

UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	


```
UU      UU      EEEEEEEEEE  TTTTTTTTTT  UU      UU      NN      NN      AAAAAA  SSSSSSSS  000000  000000
UU      UU      EEEEEEEEEE  TTTTTTTTTT  UU      UU      UU      UU      AAAAAA  SSSSSSSS  000000  000000
UU      UU      EE      EE      UU      UU      UU      UU      AA      AA      SS      SS      00      00
UU      UU      EE      EE      UU      UU      UU      UU      AA      AA      SS      SS      00      00
UU      UU      EE      EE      UU      UU      UU      UU      AA      AA      SS      SS      00      00
UU      UU      EE      EE      UU      UU      UU      UU      AA      AA      SS      SS      00      00
UU      UU      EEEEEEEEEE  TT      TT      UU      UU      UU      UU      AA      AA      SS      SS      00      00
UU      UU      EEEEEEEEEE  TT      TT      UU      UU      UU      UU      AA      AA      SS      SS      00      00
UU      UU      EEEEEEEEEE  TT      TT      UU      UU      UU      UU      AA      AA      SS      SS      00      00
UU      UU      EEEEEEEEEE  TT      TT      UU      UU      UU      UU      AA      AA      SS      SS      00      00
UU      UU      EE      EE      TT      TT      UU      UU      UU      UU      AA      AA      SS      SS      00      00
UU      UU      EE      EE      TT      TT      UU      UU      UU      UU      AA      AA      SS      SS      00      00
UU      UU      EE      EE      TT      TT      UU      UU      UU      UU      AA      AA      SS      SS      00      00
UU      UU      EE      EE      TT      TT      UU      UU      UU      UU      AA      AA      SS      SS      00      00
UUUUUUUUUU  EEEEEEEEEE  TT      TT      UUUUUUUUUU  NN      NN      AA      AA      SSSSSSSS  000000  000000
UUUUUUUUUU  EEEEEEEEEE  TT      TT      UUUUUUUUUU  NN      NN      AA      AA      SSSSSSSS  000000  000000
```

```
LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```


(2)	86	Declarations
(3)	205	Read-Only Data
(4)	469	Read/Write Data
(5)	775	RMS-32 Data Structures
(6)	829	Main Program
(10)	1128	Test the DEUNA
(16)	1917	UNA Startup Routine
(17)	1963	UNA Shutdown Routine
(18)	2004	CHECKIOSB - Check IO status block
(19)	2042	Check QIO AST Routine
(20)	2080	Remote Loopback I/O Timeout AST Routine
(21)	2143	Remote Loopback I/O Completion AST Routine
(22)	2218	Receive data AST routine
(23)	2296	DUMP Mode Routines
(24)	2365	COUNTER CHECK Routine
(25)	2497	Half Minute Timer Expiration Routine
(26)	2537	Test End Timer Expiration Routine
(27)	2573	System Service Exception Handler
(28)	2703	RMS Error Handler
(29)	2767	CTRL/C Handler
(30)	2812	Error Exit
(31)	2873	Exit Handler


```

0000 1 .TITLE UETUNAS00 VAX/VMS UETP DEVICE TEST FOR THE UNA
0000 2 .IDENT 'V04-000'
0000 3 .ENABLE SUPPRESSION
0000 4 :
0000 5 :*****
0000 6 :*
0000 7 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 :* ALL RIGHTS RESERVED.
0000 10 :*
0000 11 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 :* TRANSFERRED.
0000 17 :*
0000 18 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 :* CORPORATION.
0000 21 :*
0000 22 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :*
0000 25 :*
0000 26 :*****
0000 27 :
0000 28 :
0000 29 :++
0000 30 : FACILITY:
0000 31 : This module will be distributed with VAX/VMS under the [SYSTEST]
0000 32 : account.
0000 33 :
0000 34 : ABSTRACT:
0000 35 : This test exercises a UNA device using VMS QIO's. It is run as
0000 36 : part of the VAX/VMS UETP.
0000 37 :
0000 38 : ENVIRONMENT:
0000 39 : This program will run in user access mode, with ASTs enabled except
0000 40 : during error processing. This program requires the following
0000 41 : privileges and quotas:
0000 42 :
0000 43 : --
0000 44 :
0000 45 : AUTHOR: Larry Jones, CREATION DATE: September 03, 1982
0000 46 :
0000 47 : MODIFIED BY:
0000 48 :
0000 49 : V03-010 RNH0010 Richard N. Holstein, 08-Aug-1984
0000 50 : Extend V03-009 and prevent potential hang.
0000 51 :
0000 52 : V03-009 RNH0009 Richard N. Holstein, 11-Jul-1984
0000 53 : Make the test more robust when performing the remote loopback
0000 54 : operations - allow for the Ethernet's uncertainty by
0000 55 : retransmitting messages.
0000 56 :
0000 57 : V03-008 RNH0006 Richard N. Holstein, 15-Feb-1984

```



```

0000 58 : Take advantage of new UETP message codes. Fix SSERROR
0000 59 : interaction with RMS_ERROR. Change UETUNT$T_FILSPC field
0000 60 : to be ASCID instead of ASCIC for better error messages.
0000 61 :
0000 62 : V03-007 RNH0005 Richard N. Holstein, 03-Jan-1984
0000 63 : Bump MIN_MAXBUF to the new minimum value for the DEUNA.
0000 64 :
0000 65 : V03-006 RNH0004 Richard N. Holstein, 19-Dec-1983
0000 66 : Give correct sentinels to Test Controller. Use LIB$SIGNAL or
0000 67 : $PUTMSG throughout, instead of LIB$PUT_OUTPUT.
0000 68 :
0000 69 : V03-005 RNH0003 Richard N. Holstein, 06-Dec-1983
0000 70 : Give an error message and abort execution if the SYSGEN MAXBUF
0000 71 : parameter is too small. Change wording of CASE_FAILED message.
0000 72 :
0000 73 : V03-004 RNH0002 Richard N. Holstein, 21-Nov-1983
0000 74 : Use decimal conversion routine for unit numbers.
0000 75 :
0000 76 : V03-003 RNH0001 Richard N. Holstein, 11-Mar-1983
0000 77 : Don't signal ending message in EXIT_HANDLER.
0000 78 :
0000 79 : V03-002 LDJ0002 Larry D. Jones, 28-Feb-1983
0000 80 : Fixed error message assembly error.
0000 81 :
0000 82 : V03-001 LDJ0001 Larry D. Jones, 10-Feb-1983
0000 83 : Upgraded to conform to new template standards.
0000 84 : **

```



```

0000 86      .SBTTL Declarations
0000 87      :
0000 88      : INCLUDE FILES:
0000 89      :
0000 90      :     SYSS$LIBRARY:LIB.MLB      for general definitions
0000 91      :     SHRLIB$:UETP.MLB        for UETP definitions
0000 92      :
0000 93      :
0000 94      : MACROS:
0000 95      :
0000 96      :     $CHFDEF                    ; Condition handler frame definitions
0000 97      :     $DCDEF                    ; Device definitions
0000 98      :     $DIBDEF                   ; Device Information Block
0000 99      :     $DVIDEF                   ; $GETDVI ITMLST item codes
0000 100     :     $NMADEF                   ; Network management definition
0000 101     :     $SHRDEF                   ; Shared messages
0000 102     :     $SYIDEF                   ; $GETSYI ITMLST item codes
0000 103     :     $STSDEF                   ; Status return
0000 104     :     $UETUNTDEF                ; UETP unit block offset definitions
0000 105     :     $UETPDEF                  ; UETP
0000 106     :     $XMDEF                    ; XMDRIVER symbols
0000 107
0000 108     .MACRO TBL_ENT ENT,VALUE,STRING
0000 109         .PC1...
0000 110         .WORD <ENT@15>!VALUE
0000 111         .ADDRESS PC2...
0000 112         PC1...= .
0000 113         .PC2...
0000 114         .ASCIC /STRING/
0000 115         PC2...=.
0000 116     .ENDM TBL_ENT
0000 117
0000 118     : EQUATED SYMBOLS:
0000 119     :
0000 120     : Facility number definitions:
0000 121     :     RMS$_FACILITY = 1
0000 122
0000 123     : SHR message definitions:
0000 124     :     UETP = UETP$_FACILITY@STSS$_FAC_NO      ; Define the UETP facility code
0000 125     :
0000 126     :     UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
0000 127     :     UETP$_BEGIN = UETP!SHR$_BEGIN
0000 128     :     UETP$_ENDEDD = UETP!SHR$_ENDEDD
0000 129     :     UETP$_TEXT  = UETP!SHR$_TEXT
0000 130
0000 131     : Internal flag bits...:
0000 132     :     DUMP_MODEV = 0                      ; Set if running in dump mode
0000 133     :     ASSUME DUMP_MODEV EQ 0                ; This so we can BLBx
0000 134     :     TEST_OVERV = 1                      ; Set when test is over
0000 135     :     SAFE_TO_UPDV = 2                    ; Set when it's safe to update UETINIDEV
0000 136     :     BEGIN_MSGV = 3                     ; Set if 'BEGIN' msg has been printed
0000 137     :     FLAG_SHUTDNV = 4                   ; Set to indicate device should be
0000 138     :                                         ; shutdown if errors occur
0000 139     :     ID_FNDV = 5                          ; Remote UNA found
0000 140     :     BRDCST_NEDEDV = 6                   ; Broadcast for another UNA required
0000 141     :     LONGORSHRTV = 7                     ; Long or short report (if set then short)
0000 142

```



```

00000001 0000 143 ; ...and corresponding masks:
00000002 0000 144      DUMP_MODEM = 1@DUMP_MODEV
00000004 0000 145      TEST_OVRM  = 1@TEST_OVRV
00000008 0000 146      SAFE_TO_UPDM = 1@SAFE_TO_UPDV
00000010 0000 147      BEGIN_MSGM = 1@BEGIN_MSGV
00000020 0000 148      FLAG_SHUTDNM = 1@FLAG_SHUTDNV
00000040 0000 149      ID_FNDM    = 1@ID_FNDV
00000080 0000 150      BRDCST_NEDEDM = 1@BRDCST_NEDEDV
00000000 0000 151      LONGORSHRTM = 1@LONGORSHRTV
00000000 0000 152
00000020 0000 153 ; Miscellany:
00000028 0000 154      LC_BITM    = ^X20      ; Mask to convert lower case to upper
00000084 0000 155      REC_SIZE   = 40          ; UETINIDEV.DAT record size
00000005 0000 156      TEXT_BUFFER = 132         ; Internal text buffer size
00000006 0000 157      XMIT_EFN    = 5           ; EFN used for transmit QIO's
00000007 0000 158      RECV_EFN    = 6           ; EFN used for receive QIO's
00000004 0000 159      RDID_EFN    = 7           ; EFN used for read request
00000003 0000 160      EFN2       = 4           ; EFN used for three minute timer
0000000F 0000 161      SS_SYNCH_EFN = 3         ; Synch miscellaneous system services
0000000A 0000 162      MAX_PROC_NAME = 15        ; Longest possible process name
00000005 0000 163      MAX_DEV_DESIG = 10        ; Longest possible controller name
00000007 0000 164      MAX_UNIT_DESIG = 5         ; Longest possible unit number
00000000 0000 165      TBL_SIZE    = 7           ; Number of device counters in table
00000000 0000 166      LIN        = 0           ; Line counter ID number
00000064 0000 167      PRM        = 100         ; Test AST parameter
00000004 0000 168      RECVPOOL_SIZ = 4          ; Number of preallocated message block
00000003 0000 169      RW_TIME_ID  = 3          ; Timer ID to prevent hang when Read/write
00000001 0000 170      TIME_ID-1   = 1          ; Timer ID to prevent over all hang
00000002 0000 171      TIME_ID-2   = 2          ; Timer ID to use on REQUEST
00000050 0000 172      DIAG_BUF_SIZE = 80        ; UNA Diagnostic buffer size
000005DC 0000 173      REQUEST_BUFSIZ = 1500     ; Read request buffer size
000005DC 0000 174      MAX_UNABUF_SIZE = 1500    ; Maximum UNA transfer size
0000000E 0000 175      FORWARD_MSG_SIZE = 14     ; Size of a forward message header
00000080 0000 176      ERRCNT_CEN  = 128         ; Counters size
00000630 0000 177      MIN_MAXBUF  = 1584        ; Minimum value fo SYSGEN's MAXBUF
00000004 0000 178      RETRY_COUNT = 4           ; Count of retries for remote loopback
00000000 0000 179
0000002A 0000 180      HARD_ADR    = ^X2A        ; Diagnostic buffer offset to the
00000000 0000 181                                     ; NI hardware address
0000001A 0000 182      PCSR1       = ^X1A        ; Diagnostic buffer offset to the
00000000 0000 183                                     ; PCSR1 register contents
0000000F 0000 184      PCSR1$V_XPWR = 15          ; NI cable connected indicator
0000000E 0000 185      PCSR1$V_ICAB = 14          ; Internal cable connected indicator
00000008 0000 186      PCSR1$V_SLFTST = 8          ; Self test error code indicator
00000006 0000 187      PCSR1$S_SLFTST = 6
00000000 0000 188
00000090 0000 189      LOOP_PROTOCOL = ^X90        ; Digital loop protocol number
00000000 0000 190
00000000 0000 191 ; The following definitions are set depending on the device under test.
00000000 0000 192 ; Note that we use the UETUNT$T_FILSPC field differently from most tests. We
00000000 0000 193 ; have an ASCID data type instead of an ASCIC data type.
00000000 0000 194      DEVDEP_SIZE = 0          ; Size of device dependent part of UETUNT
000005DC 0000 195      WRITE_SIZE  = 1500        ; Size of device write buffer
000005DC 0000 196      READ_SIZE   = 1500        ; Size of device read buffer
00000000 0000 197
00000000 0000 198      PAGES = <<UETUNT$C_INDSIZ+- ; Add together all of the pieces...
00000000 0000 199                DEVDEP_SIZE+- ; ...which make up a UETP unit block...

```


UETUNAS00
V04-000

VAX/VMS UETP DEVICE TEST FOR THE UNA^{K 7}
Declarations

16-SEP-1984 01:37:49 VAX/VMS Macro V04-00
5-SEP-1984 04:26:50 [UETP.SRC]UETUNAS00.MAR;1

Page 5
(2)

00000007 0000 200
0000 201
0000 202
0000 203

WRITE SIZE+-
READ SIZE+-
511>7512>

; ...to give to the \$EXPREG service below


```

0000 205 .SBTTL Read-Only Data
0000 206 .PSECT RODATA,NOEXE,NOWRT,PAGE
0000 207
53 45 54 53 59 53 00000008'010E0000' 0000 208 ACNT_NAME: ; Process name on exit
54 000E 209 .ASCID /SYSTEST/
000F 210
41 4E 55 54 45 55 00000017'010E0000' 000F 211 TEST_NAME: ; This test name
30 30 53 001D 212 .ASCID /UETUNAS00/
0020 213
50 55 53 54 45 55 00000028'010E0000' 0020 214 SUPDEV_GBLSEC: ; How we access UETSUPDEV.DAT
56 45 44 002E 215 .ASCID /UETSUPDEV/
0031 216
41 4E 4C 52 54 43 00000039'010E0000' 0031 217 CONTROLLER: ; Logical name of controller
45 4D 003F 218 .ASCID /CTRLNAME/
0041 219
45 44 4F 4D 00000049'010E0000' 0041 220 MODE: ; Run mode logical name
0041 221 .ASCID /MODE/
004D 222
49 4E 54 53 45 54 00000055'010E0000' 004D 223 LOGNAM: ; Logical name for remote address testing
52 44 41 005B 224 .ASCID /TESTNIADR/
005E 225
00000000' 005E 226 NO_RMS_AST_TABLE: ; List of errors for which...
00000000' 0062 227 .LONG RMSS_BLN ; ...RMS cannot deliver an AST...
00000000' 0066 228 .LONG RMSS_BUSY ; ...even if one has an ERR= arg
00000000' 006A 229 .LONG RMSS_CDA ; Note that we can search table...
00000000' 006E 230 .LONG RMSS_FAB ; ...via MATCHC since <31:16>...
00000014 0072 231 .LONG RMSS_RAB ; ...pattern can't be in <15:0>
0072 232 NRAT_LENGTH = .-NO_RMS_AST_TABLE
0072 233
4E 49 24 53 59 53 0000007A'010E0000' 0072 234 SYSS$INPUT: ; Name of device from which...
54 55 50 0080 235 .ASCID /SYSS$INPUT/ ; ...the test can be aborted
0083 236
0020 004D 0083 237 INPUT_ITMLST: ; $GETDVI arg list for SYSS$INPUT
00000053'0000005B' 0083 238 .WORD 64,DVIS$ DEVNAM ; We need the equivalence name
00000000 0087 239 .LONG BUFFER,BUFFER_PTR
008F 240 .LONG 0 ; Terminate the list
0093 241
54 52 4F 50 45 52 0000009B'010E0000' 0093 242 REPORT: ; Long or short report logical name
00A1 243 .ASCID /REPORT/
00A1 244
21 20 42 58 32 21 000000A9'010E0000' 00A1 245 CS1: ; Device class and type control string
20 42 58 32 00AF 246 .ASCID /!2XB !2XB /
00B3 247
2A 20 42 58 32 21 000000BB'010E0000' 00B3 248 CS3: ; Device class-only control string
2A 00B3 249 .ASCID /!2XB **/
00C1 250
20 72 6F 72 72 45 000000CA'010E0000' 00C2 251 INIDEV_UPDERR: ; Error during exit handler
54 45 55 20 67 6E 69 74 61 64 70 75 00D0 252 .ASCID /Error updating UETINIDEV.DAT./

```



```
2E 54 41 44 2E 56 45 44 49 4E 49 00DC
                                00E7
                                00E7 253
                                00E7 254 DUMP: ; Equivalence string for MODE logical...
                                00E7 255 .ASCID /DUMP/ ; ...name to specify dump mode actions
                                00F3 256
                                00F3 257 NIADRWRONG:
                                00F3 258 .ASCID /NI address specified by logical name TESTNIADR is illegal length./

64 64 61 20 49 4E 000000FB'010E0000' 00F3
69 66 69 63 65 70 73 20 73 73 65 72 0101
61 63 69 67 6F 6C 20 79 62 20 64 65 010D
4E 54 53 45 54 20 65 6D 61 6E 20 6C 0119
65 6C 6C 69 20 73 69 20 52 44 41 49 0125
2E 68 74 67 6E 65 6C 20 6C 61 67 0131
                                013C
                                013C 259
                                013C 260 NIADRTESTING:
                                013C 261 .ASCID /Remote UNA address testing is !2XB-!2XB-!2XB-!2XB-!2XB-!2XB./

65 74 6F 6D 65 52 00000144'010E0000' 013C
73 73 65 72 64 64 61 20 41 4E 55 20 014A
20 73 69 20 67 6E 69 74 73 65 74 20 0156
32 21 2D 42 58 32 21 2D 42 58 32 21 0162
42 58 32 21 2D 42 58 32 21 2D 42 58 016E
                                017A
                                0180
                                0180 262
                                0180 263 ID_CTRSTR:
                                0180 264 .ASCID /UNA hardware address is !2XB-!2XB-!2XB-!2XB-!2XB-!2XB./

61 68 20 41 4E 55 00000188'010E0000' 0180
65 72 64 64 61 20 65 72 61 77 64 72 018E
21 2D 42 58 32 21 20 73 69 20 73 73 019A
58 32 21 2D 42 58 32 21 2D 42 58 32 01A6
2E 42 58 32 21 2D 42 58 32 21 2D 42 01B2
                                01BE
                                01BE 265
                                01BE 266 REMOTE_ID_CTR:
                                01BE 267 .ASCID /Remote UNA address found of !2XB-!2XB-!2XB-!2XB-!2XB-!2XB./

65 74 6F 6D 65 52 000001C6'010E0000' 01BE
73 73 65 72 64 64 61 20 41 4E 55 20 01CC
32 21 2D 66 6F 20 64 6E 75 6F 66 20 01D8
42 58 32 21 2D 42 58 32 21 2D 42 58 01E4
21 2D 42 58 32 21 2D 42 58 32 21 2D 01F0
                                01FC
                                0200
                                0200 268
                                0200 269 REMOTE_NOTFND:
                                0200 270 .ASCID /Remote UNA not detected on the NI./

65 74 6F 6D 65 52 00000208'010E0000' 0200
74 65 64 20 74 6F 6E 20 41 4E 55 20 020E
65 68 74 20 6E 6F 20 64 65 74 63 65 021A
                                0226
                                022A
                                022A 271
                                022A 272 BADLEN: ; Message received was wrong length
                                022A 273 .ASCID /Message #!UL received with wrong length, got !UW, expected !UW./

67 61 73 73 65 4D 00000232'010E0000' 022A
69 65 63 65 72 20 4C 55 21 23 20 65 0238
6F 72 77 20 68 74 69 77 20 64 65 76 0244
67 20 2C 68 74 67 6E 65 6C 20 67 6E 0250
65 70 78 65 20 2C 57 55 21 20 74 6F 025C
                                0268
                                0271
                                0271 274
                                0271 275 COUNTER_MSG:
                                0271 276 .ASCID /Non-zero UNA counter ''!AC'' = !UL./

65 7A 2D 6E 6F 4E 00000279'010E0000' 0271
74 6E 75 6F 63 20 41 4E 55 20 6F 72 027F
21 20 3D 20 22 43 41 21 22 20 72 65 028B
                                0297
                                029A
                                029A 277
                                029A 278 CASE_FAILED:
```



```
6F 63 65 72 6E 55 000002A2'010E0000' 029A 279 .ASCID /Unrecognized counter in NICE message./
74 6E 75 6F 63 20 64 65 7A 69 3E 67 02A8
6D 20 45 43 49 4E 20 6E 69 20 72 65 02B4
2E 65 67 61 73 73 65 02C0
02C7 280
02C7 281 TTNAME_ROPTR: ; Descriptor for recursive...
0000 003F 02C7 282 .WORD 63,0 ; ...translation of TTNAME
0000000A' 02CB 283 .ADDRESS TTNAME
02CF 284
02CF 285 CNTRLMSG:
02CF 286 .ASCID \Aborted via a user CTRL/C\
02DD
02E9
02F0 287
02F0 288 NO_CTRLNAME:
02FE 289 .ASCID /No controller specified./
030A
0310 290
0310 291 DEAD_CTRLNAME:
0310 292 .ASCID /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
031E
032A
0336
0342
034E
0357 293
0357 294 NOUNIT_SELECTED:
0357 295 .ASCID /No units selected for testing./
0365
0371
037D 296
037D 297 CAB1_UNPLUGED:
037D 298 .ASCID /WARNING the NI cable is disconnected./
038B
0397
03A3
03AA 299
03AA 300 CAB2_UNPLUGED:
03AA 301 .ASCID /The UNA inter-module cable is unplugged./
03B8
03C4
03D0
03DA 302
03DA 303 ILLEGAL_REC:
03DA 304 .ASCID /Illegal record format in file UETINIDEV.DAT!/
03E8
03F4
0400
040C
040E 305
040E 306 TIMEOUT1:
040E 307 .ASCID /One shot startup timeout./
041C
0428
042F 308
042F 309 TIMEOUT2:
```


6C 61 6D 72 6F 4E 00000437'010E0000'	042F	310	.ASCID	/Normal mode startup timeout./
75 74 72 61 74 73 20 65 64 6F 6D 20	043D			
2E 74 75 6F 65 6D 69 74 20 70	0449			
	0453	311		
	0453	312	TIMEOUT3:	
6D 20 6F 68 63 45 0000045B'010E0000'	0453	313	.ASCID	\Echo mode transmit/receive loop timeout.\
74 69 6D 73 6E 61 72 74 20 65 64 6F	0461			
6F 6F 6C 20 65 76 69 65 63 65 72 2F	046D			
2E 74 75 6F 65 6D 69 74 20 70	0479			
	0483	314		
	0483	315	TIMEOUT4:	
6D 20 6F 68 63 45 0000048B'010E0000'	0483	316	.ASCID	/Echo mode shutdown timeout./
6E 77 6F 64 74 75 68 73 20 65 64 6F	0491			
2E 74 75 6F 65 6D 69 74 20	049D			
	04A6	317		
	04A6	318	TIMEOUT5:	
6D 73 6E 61 72 54 000004AE'010E0000'	04A6	319	.ASCID	/Transmit to broadcast address startup timeout./
63 64 61 6F 72 62 20 6F 74 20 74 69	04B4			
20 73 73 65 72 64 64 61 20 74 73 61	04C0			
65 6D 69 74 20 70 75 74 72 61 74 73	04CC			
2E 74 75 6F	04D8			
	04DC	320		
	04DC	321	TIMEOUT6:	
6D 73 6E 61 72 54 000004E4'010E0000'	04DC	322	.ASCID	/Transmit to broadcast address timeout./
63 64 61 6F 72 62 20 6F 74 20 74 59	04EA			
20 73 73 65 72 64 64 61 20 74 73 61	04F6			
2E 74 75 6F 65 6D 69 74	0502			
	050A	323		
	050A	324	TIMEOUT7:	
6D 73 6E 61 72 54 00000512'010E0000'	050A	325	.ASCID	/Transmit to broadcast address shutdown timeout./
63 64 61 6F 72 62 20 6F 74 20 74 69	0518			
20 73 73 65 72 64 64 61 20 74 73 61	0524			
6D 69 74 20 6E 77 6F 64 74 75 68 73	0530			
2E 74 75 6F 65	053C			
	0541	326		
	0541	327	TIMEOUT8:	
6E 72 65 74 78 45 00000549'010E0000'	0541	328	.ASCID	/External loopback startup timeout./
20 6B 63 61 62 70 6F 6F 6C 20 6C 61	054F			
65 6D 69 74 20 70 75 74 72 61 74 73	055B			
2E 74 75 6F	0567			
	056B	329		
	056B	330	TIMEOUT9:	
77 20 67 6E 75 48 00000573'010E0000'	056B	331	.ASCID	/Hung while trying to transmit during the remote loopback test./
20 67 6E 69 79 72 74 20 65 6C 69 68	0579			
20 74 69 6D 73 6E 61 72 74 20 6F 74	0585			
72 20 65 68 74 20 67 6E 69 72 75 64	0591			
61 62 70 6F 6F 6C 20 65 74 6F 6D 65	059D			
2E 74 73 65 74 20 6B 63	05A9			
	05B1	332		
	05B1	333	TIMEOUT10:	
6D 65 73 6E 65 53 000005B9'010E0000'	05B1	334	.ASCID	/Sensemode timeout./
2E 74 75 6F 65 6D 69 74 20 65 64 6F	05BF			
	05CB	335		
	05CB	336	TIMEOUT11:	
63 20 64 61 65 52 000005D3'010E0000'	05CB	337	.ASCID	/Read counters timeout./
65 6D 69 74 20 73 72 65 74 6E 75 6F	05D9			
2E 74 75 6F	05E5			


```
55 21 20 6C 6C 41 000005F1'010E0000' 05E9
69 20 73 65 67 61 73 73 65 6D 20 4C 05E9
65 74 6F 6D 65 72 20 65 68 74 20 6E 05F7
65 74 20 6B 63 61 62 70 6F 6F 6C 20 0603
6E 61 72 74 20 65 72 65 77 20 74 73 060F
66 73 73 65 63 63 75 73 5F 21 2F 21 061B
6E 6F 20 74 75 62 20 2C 79 6C 6C 75 0627
6D 6F 73 20 72 65 74 66 61 20 79 6C 062E
61 72 74 65 72 20 65 72 65 77 20 65 063A
2E 64 65 74 74 69 6D 73 6E 0646
0652
065E
0667
0667
65 6D 20 4C 55 21 0000066F'010E0000' 0667
74 69 77 20 53 25 21 65 67 61 73 73 0675
66 6F 20 53 25 21 65 7A 69 73 20 68 0681
65 72 65 77 20 29 4C 55 35 28 23 21 068D
65 76 69 65 63 65 72 20 74 6F 6E 20 0699
72 20 65 68 74 20 79 62 5F 21 2F 21 06A5
61 62 70 6F 6F 6C 20 65 74 6F 6D 65 06A6
65 76 65 20 2C 74 73 65 74 20 6B 63 06B2
72 74 65 72 20 72 65 74 66 61 20 6E 06BE
2E 6E 6F 69 73 73 69 6D 73 6E 61 06CA
06D6
06E1
06E1
74 61 20 4C 55 21 000006E9'010E0000' 06E1
20 6F 74 20 53 25 21 74 70 6D 65 74 06EF
65 67 61 73 73 65 6D 20 64 6E 65 73 06FB
64 65 6C 69 61 66 20 4C 55 21 23 20 0707
20 2C 57 55 21 20 65 7A 69 73 20 2C 0713
2E 54 25 21 071F
0723
0723
65 74 6F 6D 65 52 0000072B'010E0000' 0723
69 74 20 6B 63 61 62 70 6F 6F 6C 20 0731
54 25 21 20 74 61 20 74 75 6F 65 6D 073D
65 67 61 73 73 65 6D 20 72 6F 66 20 0749
2E 4C 55 21 23 20 0755
075B
075B
74 63 65 70 78 45 00000763'010E0000' 075B
2C 4C 55 21 23 20 67 73 6D 20 64 65 0769
2C 57 55 21 20 68 74 67 6E 65 6C 20 0775
55 21 23 20 67 73 6D 20 74 6F 67 20 0781
55 21 20 68 74 67 6E 65 6C 20 2C 4C 078D
2E 54 25 21 20 74 61 20 57 0799
07A2
07A2
07A2
70 6F 6F 6C 6E 55 000007AA'010E0000' 07A2
20 73 65 67 61 73 73 65 6D 20 64 65 07B0
4C 55 21 23 20 68 67 75 6F 72 68 74 07BC
61 20 6E 65 65 62 20 65 76 61 68 20 07C8
2E 64 65 74 72 6F 62 07D4
```

```
338
339 RETRY_MSG: ; Indicate retransmission in rmt loop
340 .ASCID \ALL !UL messages in the remote loopback test were transmitted\
341 \!/_successfully, but only after some were retransmitted.\
342
343 LOOPBACK_FAIL_MSG: ; Couldn't loop some messages
344 .ASCID \!UL message!%S with size!%S of!#(5UL) were not received\
345 \!/_by the remote loopback test, even after retransmission.\
346
347 RETRANS_MSG: ; Retransmit for remote loopback
348 .ASCID /!UL attempt!%S to send message #!UL failed, size !UW, !%T./
349
350 LOOPBACK_TMO_MSG: ; Timeout during remote loopback
351 .ASCID /Remote loopback timeout at !%T for message #!UL./
352
353 OUT_OF_SEQ_MSG: ; Wrong or duplicate msg received
354 .ASCID /Expected msg #!UL, length !UW, got msg #!UL, length !UW at !%T./
355
356 ABORT_MSG: ; IOSB has SS$ ABORT in remote loopback
357 ; Intentional failure, msg for info only
358 .ASCID /Unlooped messages through #!UL have been aborted./
```



```
30 3D 74 69 6D 58 000007E3'010E0000' 07DB 359
57 55 21 20 3A 31 3D 76 63 65 72 2F 07DB 360 TIME_STAMP_MSG:
34 21 20 3A 68 74 67 6E 65 6C 20 2C 07E9 361 .ASCII \Xmit=0/recv=1: !UW, length: !4UW, message #!5UL, time: !%T\
20 65 67 61 73 73 65 6D 20 2C 57 55 07F5
3A 65 6D 69 74 20 2C 4C 55 35 21 23 0801
54 25 21 20 0819
081D 362
081D 363 TIME_STAMP_INTRO_MSG:
081D 364 .ASCII \IOSB is !XL, !XL; summary of recent messages follows:\
082B
0837
0843
084F
085A 365
085A 366 PASS_MSG:
085A 367 .ASCII /End of pass !UL with !UL iterations at !%D./
0868
0874
0880
088C
088D 368
088D 369 ;
088D 370 ; Table containing the identification for the names of all the
088D 371 ; UNA device counters.
088D 372 ;
088D 373
088D 374 CNTR_TBL:
088D 375 PC1... =
088D 376 .+. <TBL_SIZE*6>
0887 377 TBL_END:
0887 378
0887 379 ;
0887 380 ; *** WARNING *** TBL_SIZE must be modified to reflect the number of entries
0887 381 ; in the following table.
0887 382 ;
0887 383
0887 384 PC2... = .
0887 385 .LIST MEB
0887 386 TBL_ENT LIN,NMASC_CTLIN_RFL,<packets received in error>
0887 .=PC1...
088D .WORD <LIN@15>!NMASC_CTLIN_RFL
088F .ADDRESS PC2...
0893 .=PC2...
0887 .ASCII /packets received in error/
08C3
08CF
08B7
08D1 387 .NLIST MEB
08D1 388 TBL_ENT LIN,NMASC_CTLIN_OVR,<receives lost (internal buffer error)>
08F7 389 TBL_ENT LIN,NMASC_CTLIN_LBE,<receives lost (local buffer error)>
091A 390 TBL_ENT LIN,NMASC_CTLIN_BSM,<packets transmitted with over 1 collision>
0944 391 ; TBL_ENT LIN,NMASC_CTLIN_TFL,<transmit packets aborted>
0944 392 ; TBL_ENT LIN,NMASC_CTLIN_CDC,<transmit collision check failures>
0966 393 ; TBL_ENT LIN,NMASC_CTLIN_NPT,<no protocol type enabled>
0966 394 TBL_ENT LIN,NMASC_CTLIN_SBU,<system buffer errors>
```



```
097B 395 TBL_ENT LIN,NMA$C_CTLIN_UBU,<user buffer errors>
098E 396
098E 397 SIZE_TBL: ; Table of transfer sizes for echo mode
00000001 098E 398 .LONG 1 ; Smallest size transfer
0000002B 0992 399 .LONG 43 ; Smallest pad possible
0000002C 0996 400 .LONG 44 ; No padding done
000005DA 099A 401 .LONG 1498 ; Largest transfer with padding on
00000004 099E 402 SIZE_TBL_SIZE = <.-SIZE_TBL>/4
099E 403
099E 404 SIZE_TBL1: ; Table of transfer sizes for external
099E 405 ; loopback mode
0000002E 099E 406 .LONG 46 ; Smallest size transfer
0000007F 09A2 407 .LONG 127 ; Quarter page-1
000001FF 09A6 408 .LONG 511 ; Page-1
000005DC 09AA 409 .LONG 1500 ; Largest transfer with padding off
09AE 410 ASSUME <<.-SIZE_TBL1>/4> EQ SIZE_TBL_SIZE
09AE 411
09AE 412 TENSEC: ; 10 seconds delta time
FFFFFFF FA0A1F00 09AE 413 .LONG -10*1000*1000*10,-1
09B6 414
09B6 415 TWENSEC: ; 20 seconds delta time
FFFFFFF F4143E00 09B6 416 .LONG -10*1000*1000*20,-1
09BE 417
09BE 418 HALFMIN: ; 30 seconds delta time
FFFFFFF EE1E5D00 09BE 419 .LONG -10*1000*1000*30,-1
09C6 420
09C6 421 TWOMIN: ; Two minute delta time
FFFFFFF B8797400 09C6 422 .LONG -10*1000*1000*120,-1
09CE 423
09CE 424 THREEMIN: ; 3 minute delta time
FFFFFFF 94B62E00 09CE 425 .LONG -10*1000*1000*180,-1
09D6 426
09D6 427 UNIT_DESC: ; Descriptor used to convert unit #
00000005 09D6 428 .LONG 5
00000061' 09DA 429 .ADDRESS BUFFER+6
09DE 430
09DE 431 CONT_DESC: ; Descriptor used to convert controller...
0000 0028 09DE 432 .WORD REC_SIZE,0 ; ...from lowercase to uppercase
0000005B' 09E2 433 .ADDRESS BUFFER
09E6 434
09E6 435 FILE: ; Fills in RMS_ERR_STRING
65 6C 69 66 000009EE'010E000C' 09E6 436 .ASCID /file/
09F2 437
09F2 438 RECORD: ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 000009FA'010E0000' 09F2 439 .ASCID /recurd/
0A00 440
0A00 441 RMS_ERR_STRING: ; Announces an RMS error
41 21 20 53 4D 52 00000A08'010E0000' 0A00 442 .ASCID /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53 0A0E
44 41 21 20 65 6C 69 0A1A
0A21 443
0A21 444 PROMPT:
64 20 72 65 6C 6C 6F 72 74 6E 6F 43 0A21 445 .ASCII /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65 0A2D
20 0A39
00000019 0A3A 446 PMTSIZ = .-PROMPT
0A3A 447
```



```
20 72 6F 72 72 45 00000A42'010E0000' 0A3A 448 SENSE_ERRMSG:
64 6F 6D 20 65 73 6E 65 73 20 6E 69 0A3A 449 .ASCID /Error in sense mode test /
20 74 73 65 74 20 65 0A48
0A54
0A5B 450
0A5B 451 ERRTEST_MSG:
0A5B 452 .ASCID /Error in error test /
0A69
0A75
0A77 453
0A77 454 ASTPAR_ERRMSG:
0A77 455 .ASCID /Error in QIO AST parameter /
0A85
0A91
0A9A 456
0A9A 457 SLFTST_FAILED:
0A9A 458 .ASCID /UNA self test failed with an error code of !XL./
0AA8
0AB4
0AC0
0ACC
0AD1 459
0AD1 460 GETSYI_ITMLST: ; Check system parameters...
0AD1 461 .WORD 4,SYIS,MAXBUF ; ...to see if we have sufficient MAXBUF
0AD5 462 .ADDRESS MAXBUF,0
0ADD 463 .LONG 0 ; End of ITMLST
0AE1 464
0AE1 465 MAXBUF_MSG: ; Tell user to set adequate SYSGEN param
0AE1 466 .ASCID \The SYSGEN MAXBUF parameter is only !UL.\-
0AEF
0AFB
0B07
0B11 467 \!!_This test requires that the value be at least !UL.\
0B1D
0B29
0B35
0B41
```

104F 0004
00000000'00000C50'
00000000

59 53 20 65 68 54 00000AE9'010E0000'
20 46 55 42 58 41 4D 20 4E 45 47 53
73 69 20 72 65 74 65 6D 61 72 61 70
2E 4C 55 21 20 79 6C 6E 6F 20
73 65 74 20 73 69 68 54 5F 21 2F 21
74 20 73 65 72 69 75 71 65 72 20 74
75 6C 61 76 20 65 68 74 20 74 61 68
73 61 65 6C 20 74 61 20 65 62 20 65
2E 4C 55 21 20 74


```
00000000 0B47 469 .SBTTL Read/Write Data
00000000 470 .PSECT RWDATA,WRT,NOEXE,PAGE
0000 0000 471
0000 0000 472 TTCHAN: ; Channel associated with ctrl. term.
0000 0000 473 .WORD 0
0000 0002 474
0000 0002 475 TTNAME_RWPTR: ; Descriptor for recursive...
0000 0009' 0002 476 .WORD TTNAME_LEN,0 ; ...translation of TTNAME
0000000A' 0006 477 .ADDRESS TTNAME
000A 478 TTNAME:
54 55 50 4E 49 24 53 59 53 000A 479 .ASCII /SYS$INPUT/
00000009 0013 480 TTNAME_LEN=-TTNAME
00000049 0013 481 .BLKB 63-TTNAME_LEN
0049 482
0049 483 FLAG: ; Miscellaneous flag bits
0000 0049 484 .WORD 0 ; (See Equated Symbols for definitions)
004B 485
004B 486 FAO_BUF: ; FAO output string descriptor
0000 0084 004B 487 .WORD TEXT_BUFFER,0
0000005B' 004F 488 .ADDRESS BUFFER
0053 489
0053 490 BUFFER_PTR: ; Fake .ASCID buffer for misc. strings
0000 0084 0053 491 .WORD TEXT_BUFFER,0 ; A word for length, a word for desc.
0000005B' 0057 492 .ADDRESS BUFFER
005B 493
005B 494 BUFFER: ; FAO output and other misc. buffer
000000DF 005B 495 .BLKB TEXT_BUFFER
00DF 496
00DF 497 REQUEST_BUF:
000006BB 00DF 498 .BLKB REQUEST_BUFSIZ
06BB 499
06BB 500 DEV_DSC: ; Device name descriptor
0000 000A 06BB 501 .WORD MAX_DEV_DESIG,0
000006DA' 06BF 502 .ADDRESS DEV_NAME
06C3 503
06C3 504
06C3 505 PROCESS_NAME: ; Process name
53 41 4E 55 000006CB'010E0000' 06C3 506 .ASCII /UNAS/
06CF 507
0000000B 06CF 508 PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
000006DA 06CF 509 .BLKB PROCESS_NAME_FREE
06DA 510
000006E9 06DA 511 DEV_NAME: ; Device name buffer
0000000F 06DA 512 .BLKB MAX_DEV_DESIG+MAX_UNIT_DESIG
06E9 513 NAME_LEN = .-DEV_NAME
06E9 514
06E9 515
06E9 516
06E9 517 DIB: ; Device Information Block
0000 0074 06E9 518 .WORD DIB$K_LENGTH,0
000006F1' 06ED 519 .ADDRESS DIBBUF
06F1 520
00000765 06F1 521 .BLKB DIB$K_LENGTH
0765 522
0765 523 ERROR_COUNT: ; cumulative error count at runtime
00000000 0765 524 .LONG 0
0769 525
```


	0769	526	STATUS:		; Status value on program exit
00000000	0769	527	.LONG	0	
	076D	528			
	076D	529	QUAD_STATUS:		; IO status block for misc sys. svcs.
00000000 00000000	076D	530	.QUAD	0	
	0773	531			
	0775	532	INADDRESS:		; \$CRMPSC address storage
00000000 00000000	0775	533	.LONG	0,0	
	077D	534	OUTADDRESS:		
00000000 00000000	077D	535	.LONG	0,0	
	0785	536			
	0785	537	UNIT_NUMBER:		; Current dev unit number for GET_LINE
0000	0785	538	.WORD	0	
	0787	539			
	0787	540	DEVNAM_LEN:		; Current device name length
0000	0787	541	.WORD	0	
	0789	542			
	0789	543	RANDOM1:		; Random word #1
AAAAAAAA	0789	544	.LONG	^XAAAAAAAA	
	078D	545			
	078D	546	RANDOM2:		; Random word #2
A72EA72E	078D	547	.LONG	^XA72EA72E	
	0791	548			
	0791	549	ITERATION:		; # of times all tests were executed
00000000	0791	550	.LONG	0	
	0795	551			
	0795	552	PASS:		; Pass count
00000000	0795	553	.LONG	0	
	0799	554			
	0799	555	MSG_BLOCK:		; Auxiliary \$GETMSG info
0000079D	0799	556	.BLKB	4	
	079D	557			
	079D	558	EXIT_DESC:		; Exit handler descriptor
00000000	079D	559	.LONG	0	
000015FF	07A1	560	.ADDRESS	EXIT_HANDLER	
00000001	07A5	561	.LONG	1	
00000769	07A9	562	.ADDRESS	STATUS	
	07AD	563			
	07AD	564	ARG_COUNT:		; Argument counter used by ERROR_EXIT
00000000	07AD	565	.LONG	0	
	07B1	566			
	07B1	567	:		
	07B1	568	:	Head of self-relative UETP unit block list.	
	07B1	569	:		
	07B1	570	.	ALIGN QUAD	
	07B8	571			
	07B8	572	UNIT_LIST:		; Head of unit block circular list
00000000 00000000	07B8	573	.QUAD	0	
	07C0	574			
	07C0	575	NEW_NODE:		; Newly acquired node address
00000000 00000000	07C0	576	.QUAD	0	
	07C8	577			
	07C8	578	RSLLLEN:		
00000000	07C8	579	.LONG	0	
	07CC	580			
	07CC	581	RSLBUF:		
00000040	07CC	582	.LONG	64	

000007D4	07D0	583	.ADDRESS	64	
00000814	07D4	584	.BLKB	64	
	0814	585			
	0814	586	CVTRSL:		
00000000	0814	587	.LONG	0	
	0818	588			
	0818	589	XE_CHAN:		; UNA circuit channel
0000	0818	590	.WORD	0	
	081A	591			
	081A	592	XE_CHAN1:		; UNA circuit channel for protocol 260
0000	081A	593	.WORD	0	
	081C	594			
	081C	595	ARG_LIST:		
00000834	081C	596	.BLKL	6	
	0834	597			
	0834	598	P1BUF:		; P1 Device char buffer
00000000 00000000	0834	599	.QUAD	0	
	083C	600			
	083C	601	P2_PARAM:		
0AF1	083C	602	.WORD	NMASC_PCLI_BUS	; Buffer size
000005DC	083E	603	.LONG	MAX_UNABUF_SIZE	
	0842	604			
0451	0842	605	.WORD	NMASC_PCLI_BFN	; Buffer number
00000005	0844	606	.LONG	5	
	0848	607			
0B18	0848	608	.WORD	NMASC_PCLI_PRM	; Promiscuous mode
00000001	084A	609	.LONG	NMASC_STATE_OFF	; OFF
	084E	610			
0B19	084E	611	.WORD	NMASC_PCLI_MLT	; Multicast address state
00000001	0850	612	.LONG	NMASC_STATE_OFF	; OFF
	0854	613			
0B1B	0854	614	.WORD	NMASC_PCLI_DCH	; Data chaining
00000001	0856	615	.LONG	NMASC_STATE_OFF	; OFF
	085A	616			
0B1C	085A	617	.WORD	NMASC_PCLI_CRC	; CRC
00000000	085C	618	.LONG	NMASC_STATE_ON	; ON
	0860	619			
0B1A	0860	620	.WORD	NMASC_PCLI_PAD	; Padding
	0862	621	PAD_SWITCH:		
00000000	0862	622	.LONG	NMASC_STATE_ON	; ON
	0866	623			
0B0E	0866	624	.WORD	NMASC_PCLI_PTY	; Protocol type
	0868	625	PROTOCOL:		
00000000	0868	626	.LONG	0	; 0
	086C	627			
0B1F	086C	628	.WORD	NMASC_PCLI_EKO	; Echo mode
	086E	629	ECHO_SWITCH:		
00000000	086E	630	.LONG	NMASC_STATE_ON	; ON
	0872	631			
0456	0872	632	.WORD	NMASC_PCLI_CON	; Controller mode of
00000000	0874	633	.LONG	NMASC_LINCR_NOR	; Normal mode
	0878	634			
0B0F	0878	635	.WORD	NMASC_PCLI_MCA	; Multicast address
0008	087A	636	.WORD	8	
0001	087C	637	.WORD	NMASC_LINMC_SET	; set of
	087E	638	ADR:		
00 00 02 00 00 AB	087E	639	.BYTE	^XAB,00,00,02,00,00	; AB-00-00-02-00-00 which is


```

00000048 0884 640 ; Digital remote console address
0000083C 0884 641 P2_DESC:
0884 642 .LONG .-P2_PARAM
0888 643 .ADDRESS P2_PARAM
088C 644
088C 645
088C 646 P2_PARAM1:
0AF1 088C 647 .WORD NMASC_PCLI_BUS ; Buffer size
000005DC 088E 648 .LONG MAX_UNABUF_SIZE
0892 649
0451 0892 650 .WORD NMASC_PCLI_BFN ; Buffer number
00000005 0894 651 .LONG 5
0898 652
0B18 0898 653 .WORD NMASC_PCLI_PRM ; Promiscuous mode
00000001 089A 654 .LONG NMASC_STATE_OFF ; OFF
089E 655
0B19 089E 656 .WORD NMASC_PCLI_MLT ; Multicast address state
00000001 08A0 657 .LONG NMASC_STATE_OFF ; OFF
08A4 658
0B1B 08A4 659 .WORD NMASC_PCLI_DCH ; Data chaining
00000001 08A6 660 .LONG NMASC_STATE_OFF ; OFF
08AA 661
0B1C 08AA 662 .WORD NMASC_PCLI_CRC ; CRC
00000000 08AC 663 .LONG NMASC_STATE_ON ; ON
08B0 664
0B1A 08B0 665 .WORD NMASC_PCLI_PAD ; Padding
00000000 08B2 666 .LONG NMASC_STATE_ON ; ON
08B6 667
0B0E 08B6 668 .WORD NMASC_PCLI_PTY ; Protocol type
00000260 08B8 669 .LONG ^X260 ; 60-02
08BC 670
0B1F 08BC 671 .WORD NMASC_PCLI_EKO ; Echo mode
00000000 08BE 672 .LONG NMASC_STATE_ON ; ON
08C2 673
0456 08C2 674 .WORD NMASC_PCLI_CON ; Controller mode of
00000000 08C4 675 .LONG NMASC_LINCN_NOR ; Loopback mode
08C8 676
0B0F 08C8 677 .WORD NMASC_PCLI_MCA ; Multicast address
0008 08CA 678 .WORD 8
0001 08CC 679 .WORD NMASC_LINMC_SET ; set of
00 00 02 00 00 AB 08CE 680 .BYTE ^XAB,00,00,02,00,00 ; AB-00-00-02-00-00 which is
08D4 681 ; Digital remote console address
00000048 08D4 682 P2_DESC1:
0000088C 08D8 683 .LONG .-P2_PARAM1
08DC 684 .ADDRESS P2_PARAM1
08DC 685
000008EA 08DC 686 ADR1: ; Read QIO address to fill in
08DC 687 .BLKB 14
08EA 688
000008F8 08EA 689 ADR2: ; Remote UNA hardware address
08EA 690 .BLKB 14
08F8 691
0000 08F8 692 BRDCST_BUF:
0002 08FA 693 .WORD 0 ; Skip
08FC 694 .WORD 2 ; Function = Forward
00000902 08FC 695 SELF_ADR:
08FC 696 .BLKB 6 ; Address of self

```


0001	0902	697	.WORD	1	; Function = Reply
4A4C	0904	698	.WORD	A/LJ/	; Receipt number
00000926	0906	699	.BLKB	46-<.-BRDCST_BUF>	; Padding
0000002E	0926	700	BRDCST_BUF_LEN = .-BRDCST_BUF		
	0926	701			
	0926	702	SENSE_P1BUF:		
00000000	0926	703	.QUAD	0	; P1 buffer for sense mode test
	092E	704			
	092E	705	SENSE_P2DESC:		
00000090	092E	706	.LONG	SENSE_P2LEN	; P2 buffer descrip for sense mode test
00000936	0932	707	.ADDRESS	SENSE_P2BUF	
	0936	708			
	0936	709	SENSE_P2BUF:		
000009C6	0936	710	.BLKW	<3*24>	; P2 buffer for sense mode test
00000090	09C6	711	SENSE_P2LEN = .-SENSE_P2BUF		
	09C6	712			
	09C6	713	ERRTST_P2DESC:		
00000008	09C6	714	.LONG	ERRTST_P2LEN	; P2 desc for error test
000009CE	09CA	715	.ADDRESS	ERRTST_P2BUF	
	09CE	716			
	09CE	717	ERRTST_P2BUF:		
000009D6	09CE	718	.BLKQ	1	; P2 buffer for error test
00000008	09D6	719	ERRTST_P2LEN = .-ERRTST_P2BUF		
	09D6	720			
	09D6	721	ERRCOUNT_DESC:		
00000080	09D6	722	.LONG	ERRCNT_LEN	; Error counter buffer descrip
000009DE	09DA	723	.ADDRESS	ERRCNT_BUF	
	09DE	724			
	09DE	725	ERRCNT_BUF:		
00000BDE	09DE	726	.BLKL	ERRCNT_LEN	; Buffer for error counters
	0BDE	727			
	0BDE	728	COUNTER:		
00000000	0BDE	729	.LONG	0	; Counter value variable
	0BE2	730			
	0BE2	731	END_ADR:		
00000000	0BE2	732	.LONG	0	; Counter table end address
	0BE6	733			
	0BE6	734	XE_IOSB:		
00000BEE	0BE6	735	.BLKQ	1	; QIO IO status block for transmit
	0BEE	736			
	0BEE	737	RCV_IOSB:		
00000BF6	0BEE	738	.BLKQ	1	; QIO Io status block for receive
	0BF6	739			
	0BF6	740	SAVE_STAT:		
0000	0BF6	741	.WORD	0	; Set aside status storage
	0BF8	742			
	0BF8	743	XMIT_BUF_ADR:		
00000000	0BF8	744	.LONG	0	; Transmit buffer pointer
	0BFC	745			
	0BFC	746	RCV_BUF_ADR:		
00000000	0BFC	747	.LONG	0	; Receive buffer pointer
	0C00	748			
	0C00	749	DIAG_BUF:		
00000C50	0C00	750	.BLKB	DIAG_BUF_SIZE	
	0C50	751			
	0C50	752	MAXBUF:		
00000C54	0C50	753	.BLKL	1	; Value of the SYSGEN MAXBUF param


```

00000C58 0C54 754
          0C54 755 LOOPBACK_ACK:           ; Flag bytes to signal if some...
          0C54 756 .BLKB SIZE_TBL_SIZE    ; ...message was transmitted OK...
          0C58 757                               ; ...during the remote loopback test
          0C58 758
          0C58 759 SBP:                     ; Pointer to current slot in...
00000C5C' 0C58 760 .ADDRESS STAMP_BUFFER   ; ...STAMP_BUFFER
          0C5C 761
          0C5C 762 STAMP_BUFFER:           ; Circular list in which to keep time...
          0C5C 763                               ; ...stamps for remote loopback msgs
          0C5C 764 .REPEAT 16              ; We will have this many "slots", i.e.,
          0C5C 765                               ; a history of this many time stamps
          0C5C 766 ; Note that xxxx_STAMP have dependencies built into them for the
          0C5C 767 ; contents and length of the slots defined here.
          0C5C 768 .BLKW                    ; Transmit/receive flag
          0C5C 769 .BLKW                    ; Message length
          0C5C 770 .BLKL                    ; Message sequence number (ITERATION)
          0C5C 771 .BLKQ                    ; Time stamp
          0C5C 772 .ENDR                    ; End of slot definition
00000100 0D5C 773 TSCLL = .-STAMP_BUFFER  ; Time stamp circular list length

```



```

0D5C 775 .SBTTL RMS-32 Data Structures
0D5C 776 .ALIGN LONG
0D5C 777
0D5C 778 SYSIN_FAB: ; Allocate FAB for SYSS$INPUT
0D5C 779 $FAB-
0D5C 780 FNM = <SYSS$INPUT>
0DAC 781
0DAC 782 SYSIN_RAB: ; Allocate RAB for SYSS$INPUT
0DAC 783 $RAB-
0DAC 784 FAB = SYSIN_FAB,-
0DAC 785 ROP = PMT,-
0DAC 786 PBF = PROMPT,-
0DAC 787 PSZ = PMTSIZ,-
0DAC 788 UBF = DEV_NAME,-
0DAC 789 USZ = NAME_LEN
0DF0 790
0DF0 791 INI_FAB: ; Allocate FAB for UETINIDDEV
0DF0 792 $FAB-
0DF0 793 FAC = <GET,PUT,UPD>,-
0DF0 794 RAT = CR,-
0DF0 795 SHR = <GET,PUT,UPI>,-
0DF0 796 FNM = <UETINIDDEV.DAT>
0E40 797
0E40 798 INI_RAB: ; Allocate RAB for UETINIDDEV
0E40 799 $RAB-
0E40 800 FAB = INI_FAB,-
0E40 801 RBF = BUFFER,-
0E40 802 UBF = BUFFER,-
0E40 803 USZ = REC_SIZE
0E84 804
00000E8A 0E84 805 DDB_RFA: ; RFA storage for INI_RAB
0E84 806 .BLKB 6
0E8A 807
0E8A 808 .ALIGN LONG
0E8C 809 SUP_FAB: ; Allocate FAB for UETSUPDEV
0E8C 810 $FAB-
0E8C 811 FAC = GET,-
0E8C 812 SHR = <UPI,GET>,-
0E8C 813 RAT = CR,-
0E8C 814 FOP = UFO,-
0E8C 815 FNM = <UETSUPDEV.DAT>
0EDC 816
0EDC 817 ;
0EDC 818 ; Dummy FAB and RAB to copy to the UETP unit blocks
0EDC 819 ; The following FAB and RAB must be contiguous and in this order!
0EDC 820 ;
0EDC 821
0EDC 822 DUMMY_FAB:
0EDC 823 $FAB
0F2C 824
0F2C 825 DUMMY_RAB:
0F2C 826 $RAB RSZ = WRITE_SIZE,-
0F2C 827 USZ = READ_SIZE

```



```
0F70 829 .SBTTL Main Program
00000000 830 .PSECT UNAS,EXE,NOWRT,PAGE
0000 831
0000 832 .DEFAULT DISPLACEMENT,WORD
0000 833
0000 834 ;+
0000 835 ;
0000 836 ;
0000 837 ;
0000 838 ;
0000 839 ;-
0000 840
0000 841 .ENTRY UETUNAS00,^M<> ; Entry mask
0002 842
6D 13F6'CF DE 0002 843 MOVAL SSERROR,(FP) ; Declare exception handler
0007 844 $SETSFMS ENBFLG = #1 ; Enable system service failure mode
0010 845 $DCLEXH_S DESBLK = EXIT_DESC ; Declare an exit handler
001B 846
001B 847 $OPEN FAB = SYSIN FAB,- ; Open SYS$INPUT
001B 848 ERR = RMS_ERROR
002A 849 $CONNECT RAB = SYSIN RAB,- ; Connect RAB to SYS$INPUT
002A 850 ERR = RMS_ERROR
0039 851 BBC S^#DEV$V TRM,- ; BR if SYS$INPUT is NOT a terminal
003B 852 SYSIN FAB+FAB$L DEV,10$
003F 853 $STRNLOG_S LOGNAM = CONTROLLER,- ; Allow terminal user to specify...
003F 854 RSLLEN = DEVNAM_LEN,- ; ...a logical name...
003F 855 RSLBUF = DEVVSC ; ...for the controller to test
0058 856 CMPL RO,#SS$ NORMAL ; Was a controller specified?
005F 857 BEQL PROC_CONT_NAME ; BR if it was - go process it
0061 858 10$:
0061 859 $GET RAB = SYSIN RAB,- ; Read SYS$INPUT...
0061 860 ERR = RMS_ERROR ; ...for the controller name
0070 861 MOVW SYSIN RAB+RAB$W_RSZ,- ; Save the name length
0074 862 DEVNAM_LEN
0077 863 BNEQ PROC_CONT_NAME ; BR if we got something
0079 864 MOVL #SS$BADPARAM,STATUS ; Save an exit status if not
0082 865 PUSHAL NO_CTRLNAME ; Prepare for message...
0086 866 PUSHL #1 ; ...
0088 867 PUSHL #UETP$TEXT!ST$K_ERROR ; ...
008E 868 PUSHL #3 ; ...
0090 869 BRW ERROR_EXIT ; ...to tell of bad setup
0093 870
0093 871 PROC_CONT_NAME:
0093 872 MOVZWL DEVNAM_LEN,DEVVSC ; Set the device name length
009A 873 PUSHAL DEVVSC ; Make sure...
009E 874 PUSHAL DEVVSC ; ...that the specified controller...
00A2 875 CALLS #2,G^STR$UPCASE ; ...is all uppercase for later comparison
00A9 876 ADDL3 #1,DEVVSC,R2 ; Estimate the eventual...
00AF 877 ADDW2 R2,PROCESS_NAME ; ...process name length (incl. "-")
00B4 878 MOVAL PROCESS_NAME+8- ; Locate first available byte...
00B5 879 +MAX PROC_NAME- ; ...in process name handle...
00B5 880 -PROCESS_NAME_FREE,RO ; ...for device name
00B9 881 SUBL3 #PROCESS_NAME_FREE,- ; Will the device name fit...
00BB 882 R2,R1 ; ...in the remaining space?
00BD 883 BLEQ 10$ ; BR if it will
00BF 884 SUBL2 R1,RO ; Overwrite handle otherwise...
00C2 885 MOVW #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
```



```

      80 5F 8F 90 00C7 886 10$:
60 06DA'CF 06BB'CF 28 00CB 887
      7E D4 00D3 888
      000F'CF DF 00D5 889
      02 DD 00D9 890
      00741039 8F DD 00DB 891
00000000'GF 04 FB 00E1 892
      0049'CF 08 AB 00E8 893
      00ED 894
      00F8 895
      00F8 896
      66 0D9C'CF E1 00FA 897
      00FE 898
      00FE 900
      00FE 901
      00FE 902
      45 076D'CF E9 011A 903
      011F 904
      011F 905
      0130 906
      0130 907
      0130 908
      06C3'CF DF 0151 909
      01 DD 0155 910
      0074832B 8F DD 0157 911
00000000'GF 03 FB 015D 912
      0164 913 20$:
```

```

MOVB #^A/ /,(R0)+ ; Separate handle from device name
MOVC3 DEVDISC,DEV_NAME,(R0) ; Concatenate handle with device name
CLRL -(SP) ; Set the time stamp flag
PUSHAL TEST_NAME ; Set the test name
PUSHL #2 ; Push the argument count
PUSHL #UETP$_BEGIN!STSK_SUCCESS ; Set the message code
CALLS #4,G^LIB$SIGNAL ; Print the startup message
BISW2 #BEGIN MSGM,FLAG ; Set flag so we don't print it again
$SETPRN_S PRNAM = PROCESS_NAME ; Set the process name to UETDMPF00_x

BBC S^#DEV$V TRM,- ; BR if SYS$INPUT is NOT a terminal
SYSIN FAB+FAB$DEV,20$
$GETDVI_S DEVNAM = SYS$INPUT,- ; Get the name of...
EFN = #SS SYNCH EFN,- ; ...device which may abort test
ITMLST = INPUT ITMLST,-
IOSB = QUAD STATUS
BLBC QUAD STATUS,20$ ; Avoid CTRL/C handler if any error
$ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handler
CHAN = TTCHAN
$QIOW_S CHAN = TTCHAN,- ; Enable CTRL/C AST's...
FUNC = #IOS SETMODE!IOSM_CTRLCAST,-
P1 = CCASTHAND
PUSHAL PROCESS_NAME ; ...and tell the user...
PUSHL #1 ;
PUSHL #UETP$_ABORTC!STSK_SUCCESS ; ...how to abort gracefully...
CALLS #3,G^LIB$SIGNAL ; ...
```



```

0164 915 :
0164 916 : From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
0164 917 : and unit configuration and lets us know if the setup to run this test was
0164 918 : done correctly.
0164 919 :
0164 920 $OPEN FAB = INI_FAB,- ; Open file 'UETINIDEV.DAT'
0164 921 ERR = RMS_ERROR
0173 922 $CONNECT RAB = INI_RAB,- ; Connect the RAB and FAB
0173 923 ERR = RMS_ERROR
0182 924 $MGBLSC_S INADR = INADDRESS,- ; Connect to UETSUPDEV global section
0182 925 RETADR = OUTADDRESS,-
0182 926 GSDNAM = SUPDEV_GBLSEC,-
0182 927 FLAGS = #SEC$M_EXPREG
00000000'8F 50 D1 01A1 928 CMPL RO,#SS$NOSUCHSEC ; Was the section already there?
37 12 01A8 929 BNEQ 30$ ; BR if it was...
01AA 930 $OPEN FAB = SUP_FAB,- ; ...else open 'UETSUPDEV.DAT'
01AA 931 ERR = RMS_ERROR
01B9 932 $CRMPSC_S CHAN = SUP_FAB+FAB$STV,- ; Create the global section
01B9 933 INADR = INADDRESS,-
01B9 934 RETADR = OUTADDRESS,-
01B9 935 GSDNAM = SUPDEV_GBLSEC,-
01B9 936 FLAGS = #SEC$M_EXPREG!SEC$M_GBL
56 0781'CF 077D'CF C3 01E1 937 30$:
01E1 938 SUBL3 OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length
01E9 939 FIND_IT:
01E9 940 $GET RAB = INI_RAB,- ; Get the first record
01E9 941 ERR = RMS_ERROR
01F8 942 PUSHAL CONT_DESC ; Make sure...
09DE'CF DF 01FC 944 PUSHAL CONT_DESC ; ...that the controller name...
09DE'CF DF 0200 945 CALLS #2,G*STR$UPCASE ; ...is all uppercase letters
00000000'GF 02 8F 91 0207 946 CMPB #^A/D/,BUFFER ; Is this a DDB?
005B'CF 44 8F 91 020D 947 BEQL 10$ ; Go on if not
005B'CF 45 8F 91 020F 948 CMPB #^A/E/,BUFFER ; Is this the end of the file?
D2 12 0215 949 BNEQ FIND_IT ; Continue on if not
06BB'CF DF 0217 950 PUSHAL DEV$DSC ; Push device not supported message
06C3'CF DF 021B 951 PUSHAL PROCESS_NAME ; Parameters on the stack
02 DD 021F 952 PUSHL #2
00748333 8F DD 0221 953 PUSHL #UETP$_DENOSU
02 F0 0227 954 INSV #ST$K_ERROR,-
00 0229 955 #ST$V_SEVERITY,-
6E 03 022A 956 #ST$S_SEVERITY,(SP) ; Set the severity code...
0769'CF 6E D0 022C 957 MOVL (SP),STATUS ; ...and save it as the exit status
04 DD 0231 958 PUSHL #4
1349 31 0233 959 BRW ERROR_EXIT ; Exit in error
0236 960 10$:
06DA'CF 0061'CF 0787'CF 29 0236 961 CMPC DEVNAM_LEN,BUFFER+6,DEV_NAME ; Is this the right controller?
A7 12 0240 962 BNEQ FIND_IT ; BR if not
0E84'CF 0E50'CF 06 28 0242 963 MOV$3 #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the Record File Address
005F'CF 54 8F 91 024A 964 CMPB #^A/T/,BUFFER+4 ; Can we test this controller?
33 13 0250 965 BEQL FOUND_IT ; BR if we can...
0252 966 $FAO_S CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if we can't
0252 967 OUTLEN = BUFFER_PTR,-
0252 968 OUTBUF = FAO_BUF,-
0252 969 P1 = #DEV$DSC
0769'CF 00000000'8F D0 026B 970 MOVL #SS$_BADPARAM,STATUS ; Set return status
0053'CF DF 0274 971 PUSHAL BUFFER_PTR ; ...

```


Address	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419	Op420	Op421	Op422	Op423	Op424	Op425	Op426	Op427	Op428	Op429	Op430	Op431	Op432	Op433	Op434	Op435	Op436	Op437	Op438	Op439	Op440	Op441	Op442	Op443	Op444	Op445	Op446	Op447	Op448	Op449	Op450	Op451	Op452	Op453	Op454	Op455	Op456	Op457	Op458	Op459	Op460	Op461	Op462	Op463	Op464	Op465	
---------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--

55	000F'CF	9A	036B	1029	40\$:			
0017'CF	63	55	036B	1030		MOVZBL	TEST_NAME,R5	; Get the test name length
	1F	13	0370	1031		CMPC3	R5,(R3),TEST_NAME+8	; Are we the right test?
			0376	1032		BEQL	60\$	; BR if yes
	06BB'CF	DF	0378	1033	50\$:			
	06C3'CF	DF	0378	1034		PUSHAL	DEVDS	; Push device not supported message
	02	DD	037C	1035		PUSHAL	PROCESS_NAME	; Parameters on the stack
00748333	8F	DD	0380	1036		PUSHL	#2	; Push the argument count
	02	FO	0382	1037		PUSHL	#UETP\$_DENOSU	
	00		0388	1038		INSV	#STSSK_ERROR,-	
	6E		038A	1039			#STSSV_SEVERITY,-	
0769'CF	6E	DD	038B	1040			#STSS\$SEVERITY,(SP)	; Set the severity code...
	04	DD	038D	1041		MOVL	(SP),STATUS	; ...and save it as the exit status
	11E8	DD	0392	1042		PUSHL	#4	; Push the partial arg count...
		31	0394	1043		BRW	ERROR_EXIT	; ...and split this scene


```

0397 1101 60$:
0397 1102 $EXPREG_S PAGCNT = #PAGES, - ; Get a new node of demand zero memory
0397 1103 RETADR = NEW_NODE
07B8'CF 07C0'DF 5D 03A8 1104 INSQTI @NEW_NODE,UNIT_LIST ; Put the new node in the unit list
56 07C0'CF D0 03AF 1105 MOVL NEW_NODE,R6 ; Save a copy of its address
08 A6 01 90 03B4 1106 MOVB #1,DETUNT$B TYPE(R6) ; Set the structure type
01A4 8F B0 03B8 1107 MOVW #UETUNT$C INDSIZ+DEVDEP_SIZE,- ; Set the structure size
09 A6 03BC 1108 UETUNT$W SIZE(R6)
14 A6 06BB'CF D0 03BE 1109 MOVL DEVVSC,UETUNT$T FILSPC(R6) ; Set the device name size...
1C A6 DE 03C4 1110 MOVAL UETUNT$T FILSPC+8(R6),- ; ...and pointer to the name
18 A6 03C7 1111 UETUNT$T FILSPC+4(R6)
06BF'DF 06BB'CF 28 03C9 1112 MOVC3 DEVVSC,@DEVVSC+4,- ; Save the device name
1C A6 03D0 1113 MOVC3 #FAB$C B[N+RAB$C BLN,- ; Save a FAB and a RAB away
0094 8F 28 03D2 1114 DUMMY FAB,UETUNT$C FAB(R6)
0110 C6 0EDC'CF DE 03DC 1116 MOVAL UETUNT$K FAB(R6),R7 ; Save the FAB address
57 0110 C6 DE 03E1 1117 MOVAL UETUNT$K RAB(R6),R8 ; Save the RAB address
58 0160 C6 DE 03E6 1118 MOVL R7,RAB$L FAB(R8) ; Set the FAB address in the RAB
3C A8 57 D0 03EA 1119 MOVB UETUNT$T FILSPC(R6),- ; Set the FNS field in the FAB
14 A6 03ED 1120 FAB$B FNS(R7)
34 A7 03EF 1121 MOVAL UETUNT$T FILSPC+8(R6),- ; Set the FNA field in the FAB
1C A6 DE 03F2 1122 FAB$L_FNA(R7)
2C A7 03F4 1123 ;
03F4 1124 ; Set the device dependent parameters in here
03F4 1125 ;
FE8E 31 03F4 1126 BRW FOUND_IT ; Do the next UCB

```



```
.SBTTL Test the DEUNA

03F7 1128
03F7 1129
03F7 1130
03F7 1131 : Arrive here when we have the device configuration. In normal or loop forever
03F7 1132 : mode, set a timer far enough in the future such that we can do a reasonable
03F7 1133 : set of tests before the timer expires, but if our device gets hung, the
03F7 1134 : program won't waste too much time before noticing. Let one-shot mode be a
03F7 1135 : special case.
03F7 1136
03F7 1137 ALL_SET:
03F7 1138 TSTL UNIT_LIST ; Anything to test?
03FB 1139 BNEQ 10$ ; BR if yes
03FD 1140 PUSHAL NOUNIT_SELECTED ; Else set up the error message...
0401 1141 PUSHL #1 ; ...argument count...
0403 1142 PUSHL #UETP$ _TEXT!ST$K_ERROR ; ...signal name...
0409 1143 PUSHL #3 ; ...and parameter count
040B 1144 MOVL #SS$ BADPARAM,STATUS ; Set return status
0414 1145 BRW ERROR_EXIT ; ...and give up, complaining

0769'CF 00000000'8F 1168 31 0417 1146 10$: BISW2 #SAFE_TO_UPDM,FLAG ; OK safe to update UETINIDEV.DAT now
0049'CF 04 A8 0417 1147
041C 1148
041C 1149 $GETSYI_S ITMLST = GETSYI ITMLST ; Check for an adequate SYSGEN MAXBUF
0C50'CF 00000630 8F 1150 D1 0431 1150 CMPL #MIN_MAXBUF,MAXBUF ; Is it .GE. our minimum?
33 15 043A 1151 BLEQ 12$ ; BR if it is
043C 1152 $FAO_S CTRSTR = MAXBUF MSG,- ; Complain if it isn't...
043C 1153 OUTBUF = FAO BUF,-
043C 1154 OUTLEN = BUFFER_PTR,-
043C 1155 P1 = MAXBUF,-
043C 1156 P2 = #MIN_MAXBUF
0769'CF 00' 9A 0459 1157 MOVZBL S^#SS$ BADPARAM,STATUS
0053'CF 01 DF 045E 1158 PUSHAL BUFFER_PTR
01 DD 0462 1159 PUSHL #1
00741134 8F DD 0464 1160 PUSHL #UETP$ _TEXT!ST$K_SEVERE
03 DD 046A 1161 PUSHL #3
1110 31 046C 1162 BRW ERROR_EXIT ; ...and give up altogether

046F 1163 12$: $ASSIGN_S - ; Assign channel to the device
046F 1164 DEVNAM = DEVDSC,-
046F 1165 CHAN = XE_CHAN
046F 1166
0480 1167
14 50 E9 0480 1168 BLBC R0,15$ ; BR if no failure
0483 1169
0483 1170 $ASSIGN_S - ; Assign channel to the device
0483 1171 DEVNAM = DEVDSC,-
0483 1172 CHAN = XE_CHAN1
0494 1173
0494 1174 26 50 E8 0494 1174 BLBS R0,20$ ; BR if no failure
0497 1175 15$: 0497 1175 MOVL R0,STATUS ; Save the failure status
0769'CF 50 D0 0497 1176 INCL ERROR_COUNT ; Bump the error count
0765'CF D6 049C 1177 PUSHL STATUS ; Push the error code...
0769'CF DD 04A0 1178 PUSHL STATUS
0769'CF DD 04A4 1179 PUSHAL DEVDSC ; ...and the device designation...
06BB'CF DF 04A8 1180 PUSHAL TEST_NAME ; ...and the test name...
000F'CF DF 04AC 1181 PUSHL #3 ; ...and the arg count...
03 DD 04B0 1182 PUSHL #UETP$ _DEUNUS!ST$K_ERROR ; ...and the signal name...
0074819A 8F DD 04B2 1183 PUSHL #6 ; ...and the total argument count...
06 DD 04B8 1184
```


10C2	31	04BA	1185		BRW	ERROR_EXIT	; ...and bail out completely	
		04BD	1186	20\$:				
		04BD	1187		\$TRNLOG_S	LOGNAM = REPORT,-	; Get the report mode	
		04BD	1188			RSLEN = BUFFER_PTR,-		
		04BD	1189			RSLBUF = FAO_BUF		
005B'CF	20	8A	04D6	1190	BICB2	#LC_BITM,BUFFER	; Convert to upper case	
005B'CF	53	8F	91	04DB	1191	CMPB	#^A7S/,BUFFER	; Is this short report
	07	12	04E1	1192	BNEQ	30\$	; BR if not	
0049'CF	0080	8F	A8	04E3	1193	BISW2	#LONGORSHRTM,FLAG	; Set short report bit
			04EA	1194	30\$:			
			04EA	1195	\$TRNLOG_S	LOGNAM = MODE,-	; Get the run mode	
			04EA	1196		RSLEN = BUFFER_PTR,-		
			04EA	1197		RSLBUF = FAO_BUF		
0053'CF	DF	0503	1198		PUSHAL	BUFFER_PTR		
0053'CF	DF	0507	1199		PUSHAL	BUFFER_PTR		
00000000'GF	02	FB	050B	1200	CALLS	#2,G^STR\$UPCASE		
00EB'DF	00E7'CF	39	0512	1201	MATCHC	DUMP,@DUMP+4,-	; Check for running in DUMP mode	
005B'CF	0053'CF		0519	1202		BUFFER_PTR,BUFFER		
	05	12	051F	1203	BNEQ	35\$	; BR if we are not	
0049'CF	01	A8	0521	1204	BISW2	#DUMP_MODEM,FLAG	; Remember it if we are	
			0526	1205	35\$:			
005B'CF	4F	8F	91	0526	1206	CMPB	#^A/O/,BUFFER	; Is this a one shot?
	03	13	052C	1207	BEQL	40\$	; BR if not	
	0158	31	052E	1208	BRW	TIME_IT		
			0531	1209	40\$:			
							; Fall into one shot testing	


```

0531 1211 ;
0531 1212 ; Note: A startup is all that is needed to ONE SHOT test the UNA because
0531 1213 ; the driver performs an XESC_DG_STEST diagnostic function on the
0531 1214 ; UNA in a startup sequence and the PCSR1 contents are checked by
0531 1215 ; the device test to make sure that the self test function completed
0531 1216 ; successfully. The XESC_DG_STEST function performs a read write
0531 1217 ; verification of the UNA.
0531 1218 ;
0531 1219 ;
0531 1220 $SETIMR_S - ; Set up ten second timer
0531 1221 DAYTIM = TWENSEC,- ; to prevent hang
0531 1222 ASTADR = TIME_ERR_OUT,-
0531 1223 REQIDT = #TIMEOUTT
0548 1224
0548 1225 $SETSFM_S ENBFLG = #0 ; Disable system service failure mode
0551 1226
0551 1227 $QIOW_S FUNC = #IOS$ SETCHAR!IOSM_CTRL!IOSM_STARTUP,-
0551 1228 CHAN = XE_CHAN,-
0551 1229 IOSB = XE_IOSB,-
0551 1230 P2 = #P2_DESC,-
0551 1231 P6 = #DIAG_BUF
057C 1232
057C 1233 $SETSFM_S ENBFLG = #1 ; Enable system service failure mode
0585 1234
0585 1235 $CANTIM_S REQIDT = #TIMEOUT1 ; OK, no timeout
0594 1236
0594 1237 BBS #LONGORSHRTV,FLAG,60$ ; Skip output if short report
059A 1238 CLRL R2 ; Make an index
059C 1239 MOVAL DIAG_BUF+HARD_ADR,R3 ; Set address address
05A1 1240 MOVAL ARG_LIST,R4 ; Set arglist address
05A6 1241 50$:
05A6 1242 MOVZBL (R3)+,(R4)+ ; Parse to bytes
05A9 1243 AOBLS #6,R2,50$
05AD 1244
05AD 1245 $FAOL_S CTRSTR = ID_CTRSTR,-
05AD 1246 OUTBUF = FAO_BUF,-
05AD 1247 OUTLEN = BUFFER_PTR,-
05AD 1248 PRMLST = ARG_LIST
05C4 1249
05C4 1250 PUSHAL BUFFER_PTR
05C8 1251 PUSHL #^X10001
05CE 1252 PUSHL #UETP$ TEXT!STSSK_INFO
05D4 1253 CALLS #3,G^LIB$SIGNAL
05DB 1254 60$:
05DB 1255 MOVL DIAG_BUF+PCSR1,R2 ; Get the PCSR1 register contents
05E0 1256 CMPW R2,#3 ; Is it error free and ready (3)
05E3 1257 BNEQ 70$ ; BR if no
05E5 1258 BRW 100$ ; BR if yes else...
05E8 1259 70$:
05E8 1260 BBC #PCSR1$V_ICAB,R2,80$ ; ...check the internal cable connection.
05EC 1261 MOVZWL XE_IOSB,STATUS ; Set return status
05F3 1262 PUSHAL CAB2_UNPLUGED ; Push error message address
05F7 1263 PUSHL #1
05F9 1264 PUSHL #UETP$ TEXT!STSSK_WARNING ; Push message ID
05FF 1265 PUSHL #3
0601 1266 BRW ERROR_EXIT ; Untestable if internal cable unplugged
0604 1267 80$:

```


2F 52 0F	E1 0604 1268	BBC	#PCSR1\$V_XPWR,R2,90\$	; Check the external cable connection.
0765'CF	D6 0608 1269	INCL	ERROR_COUNT	; Bump the error count
037D'CF	DF 060C 1270	PUSHAL	CAB1_UNPLUGED	; Push error message address
000F0001 8F	DD 0610 1271	PUSHL	#^XF0001	
00741130 8F	DD 0616 1272	PUSHL	#UETP\$_TEXT!ST\$K_WARNING	; Push message ID
0765'CF	DD 061C 1273	PUSHL	ERROR_COUNT	; Push the error count
06C3'CF	DF 0620 1274	PUSHAL	PROCESS_NAME	
00010002 8F	DD 0624 1275	PUSHL	#^X10002	
00748020 8F	DD 062A 1276	PUSHL	#UETP\$_ERBOXPROC!ST\$K_WARNING	
00000000'GF 07	FB 0630 1277	CALLS	#7,G^LIB\$SIGNAL	; Print the error message
	0637 1278 90\$:			
08	EF 0637 1279	EXTZV	#PCSR1\$V_SLFTST,-	
06	0639 1280		#PCSR1\$S_SLFTST,-	
53 52	063A 1281		R2,R3	; Extract the self test error code
	063C 1282			
	063C 1283	\$FAO_S	CTRSTR = SLFTST_FAILED,-	
	063C 1284		OUTBUF = FAO_BUF,-	
	063C 1285		OUTLEN = BUFFER_PTR,-	
	063C 1286		P1 = R3	
	0651 1287			
0769'CF	0BE6'CF 3C 0651 1288	MOVZWL	XE_IOSB,STATUS	; Set return status
0053'CF	DF 0658 1289	PUSHAL	BUFFER_PTR	; Push error message address
01	DD 065C 1290	PUSHL	#1	
00741132 8F	DD 065E 1291	PUSHL	#UETP\$_TEXT!ST\$K_ERROR	; Push message ID
03	DD 0664 1292	PUSHL	#3	
0F16	31 0666 1293	BRW	ERROR_EXIT	; Mark as untestable
	0669 1294 100\$:			
	09EB 30 0669 1295	BSBW	CHECK_IOSB	; Check the IOSB results and exit if bad
5B 07B8'CF	000007B8'8F C1 066C 1296	ADDL3	#UNIT_LIST,UNIT_LIST,R11	; Get the unit block
02	88 0676 1297	BISB2	#UETUNT\$M_TESTABLE,-	
0B AB	0678 1298		UETUNT\$B_FLAGS(R11)	; This one is good
	067A 1299	\$DASSGN	S_CHAN = XE_CHAN1	; Deassign the extra channel
08AE	31 0686 1300	BRW	CLEAN_EXIT	; ONE SHOT is over


```
0689 1302 TIME_IT:
0689 1303
0689 1304 ;
0689 1305 ; Set up P1 device char buffer, P2 buffer is set up in Read/write section
0689 1306 ;
53 0836'CF DE 0689 1307 MOVAL P1BUF+2,R3 ; Address of device char for p1
83 05DC 8F B0 068E 1308 MOVW #MAX_UNABUF_SIZE,(R3)+ ; Maximum message length
63 02 90 0693 1309 MOVB #XMSM_CHR_LOOPB,(R3) ; Set loop back mode in char
52 AA 8F 9A 0696 1310 MOVZBL #^XAA,R2 ; Random number 1
53 2E 9A 069A 1311 MOVZBL #^X2E,R3 ; Random number 2
54 000005DC 8F D0 069D 1312 MOVL #MAX_UNABUF_SIZE,R4 ; Maximum message length
5B 07B8'CF 000007B8'8F C1 06A4 1313 ADDL3 #UNIT_LIST,UNIT_LIST,R11 ; Get the unit block address
56 01A4 CB DE 06AE 1314 MOVAL UETUNT$K_DEVDEP(R11),R6 ; Get the transmit buffer address
5A 0780 CB DE 06B3 1315 MOVAL UETUNT$K_DEVDEP+WRITE_SIZE(R11),R10 ; Get the receive buffer address
58 01A4 CB DE 06B8 1316 MOVAL UETUNT$K_DEVDEP(R11),R8 ; Xmit address
0BF8'CF 58 D0 06BD 1317 MOVL R8,XMIT_BUF_ADR ; Store xmit address
0BFC'CF 05DC C8 DE 06C2 1318 MOVAL WRITE_SIZE(R8),RCV_BUF_ADR ; Recv address
06C9 1319 10$:
52 53 C0 06C9 1320 ADDL2 R3,R2 ; Random number as data
86 52 90 06CC 1321 MOVB R2,(R6)+ ; Fill in the transmit buffer
F7 54 F5 06CF 1322 SOBGTR R4,10$ ; Branch if more bytes to be filled
06D2 1323
06D2 1324 $SETIMR_S DAYTIM = THREEMIN,- ; Set timer AST to 3 minutes
06D2 1325 ASTADR = TIME_SUC_OUT,-
06D2 1326 EFN = #EFN2
06E5 1327
06E5 1328 $STRNLOG_S LOGNAM = LOGNAM,- ; Get the TESTNIADR, if any
06E5 1329 RSLLEN = RSLLEN,-
06E5 1330 RSLBUF = RSLBUF
06FE 1331
00000000'8F 50 D1 06FE 1332 CMPL R0,#SS$_NOTRAN ; Was there any?
03 12 0705 1333 BNEQ 20$ ; BR if not
00A6 31 0707 1334 BRW 70$ ; Otherwise forget it
070A 1335 20$:
0C 07C8'CF D1 070A 1336 CMPL RSLLEN,#12 ; Was it the right length?
32 13 070F 1337 BEQL 30$ ; BR if yes
0765'CF D6 0711 1338 INCL ERROR_COUNT
00F3'CF DF 0715 1339 PUSHAL NIADR$WRONG ; Report a bad address
000F0001 8F DD 0719 1340 PUSHL #^XF0001
00741130 8F DD 071F 1341 PUSHL #UETP$TEXT!ST$K_WARNING
0765'CF DD 0725 1342 PUSHL ERROR_COUNT
06C3'CF DF 0729 1343 PUSHAL PROCESS_NAME
00010002 8F DD 072D 1344 PUSHL #^X10002
00748020 8F DD 0733 1345 PUSHL #UETP$ERBOXPROC!ST$K_WARNING
00000000'GF 07 FB 0739 1346 CALLS #7,G^LIB$SIGNAL
006D 31 0740 1347 BRW 70$
0743 1348 30$:
53 52 D4 0743 1349 CLRL R2 ; Make an index
08F0'CF DE 0745 1350 MOVAL ADR2+6,R3 ; Set address address
54 07D4'CF DE 074A 1351 MOVAL RSLBUF+8,R4 ; Set arglist address
074F 1352 40$:
0814'CF DF 074F 1353 PUSHAL CVTRSL
64 9F 0753 1354 PUSHAB (R4)
02 DD 0755 1355 PUSHL #2
00000000'GF 03 FB 0757 1356 CALLS #3,G^LIB$CVT_HTB
83 0814'CF 90 075E 1357 MOVB CVTRSL,(R3)+
54 02 C0 0763 1358 ADDL2 #2,R4
```



```

      E5 52 06 F2 0766 1359 AOBLESS #6,R2,40$
3D 0049'CF 07 E0 076A 1360 BBS #LUNGORSHRTV,FLAG,60$ ; Skip message if short report
      52 D4 0770 1361 CLRL R2 ; Make an index
      53 08F0'CF DE 0772 1362 MOVAL ADR2+6,R3 ; Set address address
      54 081C'CF DE 0777 1363 MOVAL ARG_LIST,R4 ; Set arglist address
      84 83 9A 077C 1364 50$: MOVZBL (R3)+,(R4)+ ; Parse to bytes
      F9 52 06 F2 077F 1366 AOBLESS #6,R2,50$
      0783 1367
      0783 1368 $FAOL_S CTRSTR = NIADRTSTING,- ; Format address
      0783 1369 OUTBUF = FAO BUF,-
      0783 1370 OUTLEN = BUFFER_PTR,-
      0783 1371 PRMLST = ARG_LIST
      079A 1372
      0053'CF DF 079A 1373 PUSHAL BUFFER_PTR ; Output message
      01 DD 079E 1374 PUSHL #1
      00741133 8F DD 07A0 1375 PUSHL #UETP$ TEXT+STSSK_INFO
00000000'GF 03 FB 07A6 1376 CALLS #3,G^LIB$SIGNAL
      01A1 31 07AD 1377 60$:
      07B0 1378 70$: BRW XMIT_START
      0862'CF 00 D0 07B0 1380 MOVL #NMASC_STATE_ON,PAD_SWITCH ; Set up dynamic characteristics
      086E'CF 00 D0 07B5 1381 MOVL #NMASC_STATE_ON,ECHO_SWITCH
0868'CF 00000090 8F D0 07BA 1382 MOVL #LOOP_PROTOCOL,PROTOCOL
      07C3 1383
      07C3 1384 $SETIMR_S - ; Set up half minute timer
      07C3 1385 DAYTIM = HALFMIN,- ; to prevent hang
      07C3 1386 ASTADR = TIME_ERR_OUT,-
      07C3 1387 REQIDT = #TIMEOUT2
      07DA 1388 START_CONT:
      0818'CF DF 07DA 1389 PUSHAL XE_CHAN ; Start the UNA on the first channel
      0884'CF DF 07DE 1390 PUSHAL P2_DESC
OFF8'CF 02 FB 07E2 1391 CALLS #2,UNA_STARTUP
      07E7 1392
      081A'CF DF 07E7 1393 PUSHAL XE_CHAN1 ; Start the UNA on the second channel
      08D4'CF DF 07EB 1394 PUSHAL P2_DESC1
OFF8'CF 02 FB 07EF 1395 CALLS #2,UNA_STARTUP
      07F4 1396
      0049'CF 10 A8 07F4 1397 BISW2 #FLAG_SHUTDNM,FLAG ; Set flag to say shut down the
      07F9 1398 ; device if errors occur
      07F9 1399
      07F9 1400 $CANTIM_S REQIDT = #TIMEOUT2 ; Cancel hang timer
      0808 1401
      0808 1402 MOVL DIAG_BUF+HARD_ADR,SELF_ADR ; Set up NI return address
      080F 1403 MOVW DIAG_BUF+HARD_ADR+4,-
      0813 1404 SELF_ADR+4
      0816 1405
      0816 1406 $SETIMR_S DAYTIM = TWOMIN,- ; Set timer AST to 2 minutes
      0816 1407 ASTADR = REQUEST_NOTFND,-
      0816 1408 REQIDT = #TIME_ID_2
      0829 1409
      0829 1410 $QIO_S - ; Do a read of data
      0829 1411 CHAN = XE_CHAN1,-
      0829 1412 FUNC = #IOS_READVBLK,-
      0829 1413 EFN = #RDID_EFN,-
      0829 1414 IOSB = XE_IOSB,-
      0829 1415 ASTADR = REQUEST_FND,-
```


UETUNAS00
V04-000

VAX/VMS UETP DEVICE TEST FOR THE UN⁹
Test the DEUNA

16-SEP-1984 01:37:49
5-SEP-1984 04:26:50

VAX/VMS Macro V04-00
[UETP.SRC]UETUNAS00.MAR;1

Page 34
(12)

0829 1416
0829 1417
0829 1418
085A 1419
085A 1420

P1 = REQUEST_BUF,-
P2 = #REQUEST_BUF SIZ,-
P5 = #ADR2


```

085A 1422 RESTART:
085A 1423
085A 1424 ;
085A 1425 ; Loopback test transmit and receive random data with different message length
085A 1426 ;
085A 1427
085A 1428 XMIT_LOOP:
085A 1429 $SETIMR_S - ; Set half minute timer to prevent hang
085A 1430 DAYTIM = HALFMIN,-
085A 1431 ASTADR = TIME_ERR_OUT,-
085A 1432 REQIDT = #TIMEOUT3
0871 1433
57 098E'CF D4 0871 1434 CLRL R6 ; Set an index
DE 0873 1435 MOVAL SIZE_TBL,R7 ; Get size table address
0878 1436 XMIT:
0878 1437 $QIO_S - ; Transmit data message
0878 1438 EFN = #XMIT EFN,- ; Event flag
0878 1439 CHAN = XE_CHAN,- ; Channel
0878 1440 FUNC = #IOS$ WRITEVBLK,- ; Transmit
0878 1441 IOSB = XE_IOSB,- ; IOSB
0878 1442 ASTADR = CHK_QIO_AST,- ; Completion ast routine
0878 1443 ASTPRM = #PRM,- ; Ast parameter
0878 1444 P1 = @XMIT_BUF_ADR,- ; Addr of transmit buffer
0878 1445 P2 = (R7)[R6],- ; message length in bytes
0878 1446 P5 = #DIAG_BUF+HARD_ADR ; NI address
08AA 1447
08AA 1448 $QIOW_S - ; Read data message
08AA 1449 EFN = #RECV EFN,- ; Event flag
08AA 1450 CHAN = XE_CHAN,- ; Channel
08AA 1451 FUNC = #IOS$ READVBLK,- ; Receive message
08AA 1452 IOSB = RCV_IOSB,- ; IOSB
08AA 1453 ASTADR = RECV_AST,- ; Completion ast to check data received
08AA 1454 ASTPRM = (R7)[R6],- ; Ast parameter = message length
08AA 1455 P1 = @RECV_BUF_ADR,- ; Receive buffer
08AA 1456 P2 = (R7)[R6],- ; Message length in bytes
08AA 1457 P5 = #ADR1 ; NI address
08D9 1458
00 00000BFC'9F 00 2C 08D9 1459 MOVCS #0,@#RECV_BUF_ADR,#0,-
OBFC'DF 05DC 8F 08E1 1460 #MAX_UNABUF_SIZE,@RECV_BUF_ADR ; Clear the read buffer
0791'CF D6 08E7 1461 INCL ITERATION ; Increment iteration count
89 56 04 F2 08EB 1462 AOBLS #SIZE_TBL_SIZE,R6,XMIT ; Loop for size_tbl_size times
08EF 1463
08EF 1464 $CANTIM_S - ; Cancel hang timer
08EF 1465 REQIDT = #TIMEOUT3
08FE 1466
03 0049'CF 01 E1 08FE 1467 BBC #TEST_OVERV,FLAG,10$ ; Is the test over?
0435 31 0904 1468 BRW SENSE_TEST
0907 1469 10$:
0049'CF 06 E1 0907 1470 BBC #BRDCST_NEDEDV,FLAG,-
03 090C 1471 20$ ; BR if UNA adr found
0098 31 090D 1472 BRW DO_BRDCST ; BR if no UNA adr found in 2 minute
0910 1473 ; Read request
0910 1474 20$:
03 0049'CF 05 E0 0910 1475 BBS #ID_FNDV,FLAG,30$ ; BR if UNA adr found
FF41 31 0916 1476 BRW XMIT_LOOP ; BR if no UNA adr found yet
0919 1477 30$:
0919 1478 $SETIMR_S - ; Set ten sec timer to prevent hang

```



```
0919 1479 DAYTIM = TENSEC,-
0919 1480 ASTADR = TIME_ERR_OUT,-
0919 1481 REQIDT = #TIMEOUT4
0930 1482
0818'CF DF 0930 1483 PUSHAL XE_CHAN ; Shutdown the UNA on the first channel
102C'CF 01 FB 0934 1484 CALLS #1,UNA_SHUTDOWN
081A'CF DF 0939 1485
102C'CF 01 FB 093D 1486 PUSHAL XE_CHAN1 ; Shutdown the UNA on the second channel
0942 1487 CALLS #1,UNA_SHUTDOWN
0942 1488 $CANTIM_S REQIDT = #TIMEOUT4
0951 1489
0951 1490
0951 1491 XMIT_START:
0862'CF 01 DO 0951 1492 MOVL #NMASC_STATE_OFF,PAD_SWITCH ; Turn off padding
086E'CF 01 DO 0956 1493 MOVL #NMASC_STATE_OFF,ECHO_SWITCH ; Turn off echo mode
0868'CF 00000090 8F DO 095B 1494 MOVL #LOOP_PROTOCOL,PROTOCOL ; Set to loop back protocol type
0964 1495
0964 1496 $SETIMR_S - ; Set twenty sec timer to prevent hang
0964 1497 DAYTIM = TWENSEC,-
0964 1498 ASTADR = TIME_ERR_OUT,-
0964 1499 REQIDT = #TIMEOUT5
097B 1500
0818'CF DF 097B 1501 PUSHAL XE_CHAN ; Startup a UNA channel in loopback
0884'CF DF 097F 1502 PUSHAL P2_DESC
OFF8'CF 02 FB 0983 1503 CALLS #2,UNA_STARTUP
0988 1504
08FC'CF 0C2A'CF DO 0988 1505 MOVL DIAG_BUF+HARD_ADR,SELF_ADR ; Set up NI return address
0C2E'CF BO 098F 1506 MOVW DIAG_BUF+HARD_ADR+4,-
0900'CF 0993 1507 SELF_ADR+4
0996 1508
0996 1509
0996 1510 $CANTIM_S REQIDT = #TIMEOUT5
0207 31 09A5 1511 BRW XMIT_SETUP ; Start external looping.
09A5 1512
09A8 1513 DO_BRDCST:
09A8 1514 $SETSFM_S ENBFLG = #0 ; Disable errors
09B1 1515 $CANCEL_S CHAN = XE_CHAN1 ; Give up the pending read request...
09BD 1516 $WAITFR_S EFN = #RDID_EFN ; ...and wait for it to die
0049'CF 0060 8F AA 09C6 1517 BICW2 #<ID_FNDM!BRDCST_NEDEDM>,FLAG ; Init the ID found and brdcst flag
09CD 1518 $SETSFM_S ENBFLG = #1 ; Safe to turn on errors again
09D6 1519
09D6 1520 $SETIMR_S - ; Set ten sec timer to prevent hang
09D6 1521 DAYTIM = TENSEC,-
09D6 1522 ASTADR = TIME_ERR_OUT,-
09D6 1523 REQIDT = #TIMEOUT4
09ED 1524
0818'CF DF 09ED 1525 PUSHAL XE_CHAN ; Shutdown the UNA on the first channel
102C'CF 01 FB 09F1 1526 CALLS #1,UNA_SHUTDOWN
081A'CF DF 09F6 1527
102C'CF 01 FB 09FA 1528 PUSHAL XE_CHAN1 ; Shutdown the UNA on the second channel
09FF 1529 CALLS #1,UNA_SHUTDOWN
09FF 1530
09FF 1531 $CANTIM_S REQIDT = #TIMEOUT4
0A0E 1532
08EA'CF FFFFFFFF 8F DO 0A0E 1533 MOVL #-1,ADR2 ; UNA address to broadcast
08EE'CF FFFF 8F BO 0A17 1534 MOVW #-1,ADR2+4
```



```
0862'CF 01 DO 0A1E 1536
086E'CF 01 DO 0A23 1537
C868'CF 00000090 8F DO 0A28 1538
                                0A31 1539
                                0A31 1540
                                0A31 1541
                                0A31 1542
                                0A31 1543
                                0A48 1544
                                0A48 1545
                                0A4C 1546
OFF8'CF 02 FB 0A50 1547
                                0A55 1548
                                0A55 1549
                                0A64 1550
                                0A64 1551
                                0A6D 1552
                                0A6D 1553
                                0A6D 1554
                                0A6D 1555
                                0A6D 1556
                                0A84 1557
                                0A84 1558
                                0A84 1559
                                0A84 1560
                                0A84 1561
                                0A84 1562
                                0A84 1563
                                0A84 1564
                                0A84 1565
                                0AB5 1566
                                0AB5 1567
                                0AC4 1568
                                0AC4 1569
                                0AC4 1570
                                0AC4 1571
                                0AD7 1572
                                0AD7 1573
                                0AD7 1574
                                0AD7 1575
                                0AD7 1576
                                0AD7 1577
                                0AD7 1578
                                0AD7 1579
                                0AD7 1580
                                0B0C 1581
OBF6'CF OBE6'CF B0 0B0C 1582
                                0B13 1583
                                0B13 1584
                                0B13 1585
                                0B13 1586
                                0B13 1587
                                0B2A 1588
                                0B2A 1589
102C'CF 01 FB 0B2E 1590
                                0B33 1591
                                0B33 1592
```

```
MOVL #NMASC_STATE_OFF,PAD_SWITCH ; Turn off padding
MOVL #NMASC_STATE_OFF,ECHO_SWITCH ; Turn off echo mode
MOVL #LOOP_PROTOCOL,PROTOCOL ; Set to loop back protocol type

$SETIMR_S - ; Set twenty sec timer to prevent hang
    DAYTIM = TWENSEC,-
    ASTADR = TIME_ERR_OUT,-
    REQIDT = #TIMEOUT5

PUSHAL XE_CHAN ; Startup a UNA channel in loopback
PUSHAL P2_DESC
CALLS #2,UNA_STARTUP

$SCANTIM_S REQIDT = #TIMEOUT5

$SETSFM_S ENBFLG = #0 ; If not don't error hard

$SETIMR_S - ; Set ten sec timer to prevent hang
    DAYTIM = TENSEC,-
    ASTADR = TIME_ERR_OUT,-
    REQIDT = #TIMEOUT6

$QIO_S CHAN = XE_CHAN,- ; Xmit to the broadcast address
    FUNC = #IOS$WRITEVBLK,-
    IOSB = XE_IOSB,-
    ASTADR = CHK_QIO_AST,- ; Completion ast routine
    ASTPRM = #PRM,- ; Ast parameter
    P1 = BRDCST_BUF,- ; Addr of transmit buffer
    P2 = #BRDCST_BUF_LEN,- ; message length in bytes
    P5 = #ADR2 ; NI broadcast address

$SCANTIM_S REQIDT = #TIMEOUT6

$SETIMR_S DAYTIM = TENSEC,- ; Should get a response in ten seconds
    ASTADR = ABRT_BRDCST,-
    REQIDT = #TIME_ID_2

$QIOW_S CHAN = XE_CHAN,- ; Receive from the broadcast address
    FUNC = #IOS$READVBLK,-
    IOSB = XE_IOSB,-
    ASTADR = REQUEST_FND,- ; Completion ast routine
    ASTPRM = #PRM,- ; Ast parameter
    P1 = @RECV_BUF_ADR,- ; Addr of receive buffer
    P2 = #MAX_ONABOF_SIZE,- ; message length in bytes
    P5 = #ADR2 ; NI address

MOVW XE_IOSB,SAVE_STAT ; Save the read status

$SETIMR_S - ; Set ten sec timer to prevent hang
    DAYTIM = TENSEC,-
    ASTADR = TIME_ERR_OUT,-
    REQIDT = #TIMEOUT7

PUSHAL XE_CHAN ; Shutdown the UNA
CALLS #1,UNA_SHUTDOWN

$SCANTIM_S REQIDT = #TIMEOUT7
```


0A 0BF6'CF	E8	0B42	1593		
0862'CF 00	DO	0B42	1594	BLBS	SAVE_STAT,10\$; BR if read made it
086E'CF 00	DO	0B47	1595	MOVL	#NMA\$C_STATE_ON,PAD_SWITCH ; Turn padding back on
		0B4C	1596	MOVL	#NMA\$C_STATE_ON,ECHO_SWITCH ; Turn echo back on
		0B51	1597		
		0B51	1598		
		0B51	1599	\$SETIMR_S -	; Set twenty sec timer to prevent hang
		0B51	1600	DAYTIM = TWENSEC,-	
		0B51	1601	ASTADR = TIME_ERR_OUT,-	
		0B51	1602	REQIDT = #TIMEOUT8	
		0B68	1603		
0B18'CF	DF	0B68	1604	PUSHAL	XE_CHAN ; Restart the UNA
0884'CF	DF	0B6C	1605	PUSHAL	P2_DESC
OFF8'CF 02	FB	0B70	1606	CALLS	#2,UNA_STARTUP
		0B75	1607		
		0B75	1608	\$CANTIM_S REQIDT = #TIMEOUT8	
		0B84	1609		
26 0BF6'CF	E8	0B84	1610	BLBS	SAVE_STAT,XMIT_SETUP ; If we got one then start external loopback
		0B89	1611		
0200'CF	DF	0B89	1612	PUSHAL	REMOTE_NOTFND ; Else report an error
01	DD	0B8D	1613	PUSHL	#1
00741130 8F	DD	0B8F	1614	PUSHL	#UETP\$TEXT!ST\$K_WARNING
00000000'GF 03	FB	0B95	1615	CALLS	#3,G^LIB\$SIGNAL
		0B9C	1616		
		0B9C	1617	\$SETSFMS ENBFLG = #1	; It's OK now
0049'CF 0040 8F	AA	0BA5	1618	BICW2	#BRDCST_NEDEDM,FLAG ; We're giving up
FCAB	31	0BAC	1619	BRW	XMIT_LOOP ; Carry on in Echo mode


```
OBAF 1621 :
OBAF 1622 : When requesting looped back messages from a remote node, we have to account
OBAF 1623 : for the Ethernet's uncertainty: messages are not guaranteed to reach their
OBAF 1624 : destination and it is the responsibility of the sender to retransmit them
OBAF 1625 : if they do not. At the same time, UETP must terminate. We therefore limit
OBAF 1626 : the number of attempts we will make at retransmission and the time we will
OBAF 1627 : wait for any one attempt to transmit. Any I/O is $CANCELLED after it
OBAF 1628 : completes or after we time out; if it were to complete later in the test it
OBAF 1629 : could give us inconsistent results.
OBAF 1630 :
OBAF 1631 :
OBAF 1632 XMIT_SETUP:
OBAF 1633   MOV C3, #FORWARD_MSG_SIZE, - ; Put the forward message header in...
OBB1 1634   BRDCST BUF, @XMIT_BUF_ADR ; ...the transmit buffer
OBB7 1635   MOV C5, #0, #0, #0, #ISCLL, - ; Clear out time stamps from previous...
OBBF 1636   STAMP_BUFFER ; ...run (needed for LOOP mode)
OBC2 1637
OBC2 1638 REMOTE_LOOPBACK_TEST:
OBC2 1639   MOV C5, #0, #0, #0, #SIZE_TBL_SIZE, - ; Clear flags which say that some...
OBC8 1640   LOOPBACK_ACK ; ...transmission size succeeded
OBCB 1641   CLRQ R6 ; Initialize counter for msg sizes...
OBCD 1642   ; ...and retransmission flag
OBCD 1643   MOVAL SIZE_TBL1, R8 ; Point to table of message sizes
OBD2 1644   MOVZWL XE_CHAN, R9 ; Stick channel in convenient holder
OBD7 1645   $SETIMR_S DAYTIM = HALFMIN, - ; Allow time for messages to complete
OBD7 1646   ASTADR = LOOPBACK_TMO, -
OBD7 1647   REQIDT = #LOOPBACK_TMO
OBE2 1648 10$: INCL ITERATION ; Count the messages we're to receive
OBF2 1649   CLRL R10 ; Limits retransmission attempts
OBF4 1650   MOV C5, #0, #0, #0, #MAX_UNABUF_SIZE, - ; Clear buffer into which...
OBF4 1651   @RCV_BUF_ADR ; ...messages are read
OBF4 1652   $QIO_S EFN = #RCV_EFN, - ; Be ready to read if the msg arrives
OBF4 1653   CHAN = R9, -
OBF4 1654   FUNC = #IOS_READLBLK, -
OBF4 1655   IOSB = RCV_IOSB, -
OBF4 1656   ASTADR = LOOPBACK_AST, -
OBF4 1657   ASTPRM = R6, -
OBF4 1658   P1 = @RCV_BUF_ADR, -
OBF4 1659   P2 = (R8), -
OBF4 1660   P5 = #ADR1 ; NI address
OBF4 1661
OBF4 1662 BLBC FLAG, 20$ ; Skip sequence number if normal running
OBF4 1663   MOVL XMIT_BUF_ADR, R11 ; In dump mode, use sequence number...
OBF4 1664   MOVL ITERATION, 10(R11) ; ... (ITERATION) to verify message
OBF4 1665 20$: PUSHL 10(R11)
OBF4 1666   PUSHL (R8)
OBF4 1667   CALLS #2, XMIT_STAMP ; In dump mode, keep track of messages
OBF4 1668   $QIO_S EFN = #XMIT_EFN, - ; Tell some node to loop back a message
OBF4 1669   CHAN = R9, -
OBF4 1670   FUNC = #IOS_WRITEBLK, -
OBF4 1671   IOSB = XE_IOSB, -
OBF4 1672   P1 = @XMIT_BUF_ADR, -
OBF4 1673   P2 = (R8), -
OBF4 1674   P5 = #ADR2+6 ; Remote NI address
OBF4 1675   $WAITFR_S EFN = #RCV_EFN ; Wait for message to be returned
OBF4 1676   BLBS LOOPBACK_ACK(R6), 30$ ; BR if this size message got through
OBF4 1677   MOVL #1, R7 ; Indicate retransmission
```

0100 8F 00 00 0E 28 OBAF 1633
OBF8'DF 00 8F 00 OBB1 1634
0C5C'CF 2C OBB7 1635
OBC2 1637
04 00 00 8F 00 2C OBC2 1638
0C54'CF 56 7C OBC8 1640
58 099E'CF DE OBCB 1641
59 0818'CF 3C OBCD 1642
0791'CF D6 OBE2 1648
5A D4 OBF2 1649
05DC 8F 00 00 8F 00 2C OBF4 1650
OBF4 1651
OBF4 1652
OBF4 1653
OBF4 1654
OBF4 1655
OBF4 1656
OBF4 1657
OBF4 1658
OBF4 1659
OBF4 1660
OBF4 1661
OB 0049'CF E9 OBF4 1662
5B OBF8'CF D0 OBF4 1663
OA AB 0791'CF D0 OBF4 1664
OA AB DD OBF4 1665
68 DD OBF4 1666
121B'CF 02 FB OBF4 1667
OBF4 1668
OBF4 1669
OBF4 1670
OBF4 1671
OBF4 1672
OBF4 1673
OBF4 1674
OBF4 1675
38 0C54'C6 E8 OBF4 1676
57 01 D0 OBF4 1677

20	0049'CF	5A	D6	OC7C	1678		\$CLREF_S	EFN = #RECV_EFN		; Flag was set by timeout routine
			E9	OC85	1679		INCL	R10		; Figure which retransmission is next
				OC87	1680		BLBC	FLAG,25\$		; Skip message if normal running
				OC8C	1681		\$FAO_S	CTRSTR = RETRANS MSG,-		
				OC8C	1682			OUTLEN = BUFFER_PTR,-		
				OC8C	1683			OUTBUF = FAO_BUF,-		
				OC8C	1684			P1 = R10,-		
				OC8C	1685			P2 = ITERATION,-		
				OC8C	1686			P3 = (R8),-		
				OC8C	1687			P4 = #0		
	0611	30	OCA9	1688		BSBW	DUMP SIGNAL			
5A	04	D1	OCAC	1689	25\$:	CMPL	#RETRY_COUNT,R10		; Loop to re-xmit...	
	89	14	OCAF	1690		BGTR	20\$		; ...if any attempts left	
	88	D5	OCB1	1691	30\$:	TSTL	(R8)+		; Point to the next message size	
FF35	56	01	F1	OCB3	1692	ACBL	#SIZE_TBL_SIZE-1,#1,R6,10\$		; Loop while there are more msg sizes	
	46	57	E9	OCB9	1693	BLBC	R7,70\$		; Skip msg if no retransmissions	
	5A	5E	D0	OCBC	1694	MOVL	SP,R10		; Save original SP for later cleanup	
	56	7C	OCBF	1695		CLRQ	R6		; Counter for msg sizes which failed...	
			OCC1	1696					; ... (R7) and loop index (R6)	
	07	OC54'C6	E8	OCC1	1697	40\$:	BLBS	LOOPBACK_ACK(R6),50\$		; BR if this size message got across
	57	D6	OCC6	1698		INCL	R7		; Count size(s) which failed	
	099E'CF46	DD	OCC8	1699		PUSHL	SIZE_TBL1[R6]		; Save the message size	
FO	56	04	F2	OCCD	1700	50\$:	AOBLSS	#SIZE_TBL_SIZE,R6,40\$		; Loop through remainder of flags
58	05E9'CF	DE	OCD1	1701		MOVAL	RETRY_MSG,R8		; Assume we want msg for all sizes OK	
	04	DD	OCD6	1702		PUSHL	#SIZE_TBL_SIZE		; Save total count of message sizes	
	57	D5	OCD8	1703		TSTL	R7		; Did all msg sizes eventually make it?	
	0A	13	OCDA	1704		BEQL	60\$		; BR if they did	
58	0667'CF	DE	OCDC	1705		MOVAL	LOOPBACK_FAIL_MSG,R8		; Use other message if they did not	
	6E	57	D0	OCE1	1706	MOVL	R7,(SP)		; Finally, save count of failed msgs...	
	57	DD	OCE4	1707		PUSHL	R7		; ...for !%S and !# both	
	59	5E	D0	OCE6	1708	60\$:	MOVL	SP,R9		; Save pointer to \$FAO args
			OCE9	1709		\$FAOL_S	CTRSTR = (R8),-		; Form msg indicating fails or attempts	
			OCE9	1710			OUTBUF = FAO_BUF,-			
			OCE9	1711			OUTLEN = BUFFER_PTR,-			
			OCE9	1712			PRMLST = (R9)			
	05BE	30	OCFC	1713		BSBW	DUMP SIGNAL		; Tell the user of our difficulties	
5E	5A	D0	OCFF	1714		MOVL	R10,SP		; Clean \$FAOL args from the stack	
			OD02	1715	70\$:	\$CANCEL_S	CHAN = XE_CHAN		; We can't afford to get old msgs later	
			OD0E	1716		\$CANTIM_S	REQIDT = #LOOPBACK_TMO		; Now we know test won't hang	
			OD1D	1717						
19	0049'CF	01	E0	OD1D	1718	BBS	#TEST_OVERV,FLAG,SENSE_TEST		; Is the test over?	
	FE9C	31	OD23	1719		BRW	REMOTE_LOOPBACK_TEST		; Continue in external looping	


```
0049'CF 0040 8F 0000 0D26 1721 :
0049'CF 0040 8F 0000 0D26 1722 : If in ten seconds the transmit to the broadcast address does not succeed
0049'CF 0040 8F 0000 0D26 1723 : either the device is broken or there is no one else on the NI so abort
0049'CF 0040 8F 0000 0D26 1724 : the read request and continue looping in Echo mode.
0049'CF 0040 8F 0000 0D26 1725 :
0049'CF 0040 8F 0000 0D26 1726 :
0049'CF 0040 8F 0000 0D26 1727 ABRT_BRDCST:
0049'CF 0040 8F 0000 0D26 1728 .WORD 0
0049'CF 0040 8F 0000 0D28 1729 BISR2 #BRDCST_NEDEDM,FLAG ; Signal the CANCELED AST routine
0049'CF 0040 8F 0000 0D2F 1730 $CANCEL_S CHAN = XE_CHAN ; Give up completely
0049'CF 0040 8F 0000 0D3B 1731 RET
0049'CF 0040 8F 0000 0D3C 1732
0049'CF 0040 8F 0000 0D3C 1733 SENSE_TEST:
0049'CF 0040 8F 0000 0D3C 1734
0049'CF 0040 8F 0000 0D3C 1735 $SETIMR_S - ; Set ten sec timer to prevent hang
0049'CF 0040 8F 0000 0D3C 1736 DAYTIM = TENSEC,-
0049'CF 0040 8F 0000 0D3C 1737 ASTADR = TIME_ERR_CUT,-
0049'CF 0040 8F 0000 0D3C 1738 REQIDT = #TIMEOUT10
0049'CF 0040 8F 0000 0D53 1739
0049'CF 0040 8F 0000 0D53 1740 $QIOW_S - ; Read device characteristics
0049'CF 0040 8F 0000 0D53 1741 CHAN = XE_CHAN,-
0049'CF 0040 8F 0000 0D53 1742 FUNC = #IOS$ SENSEMODE!IOSM_CTRL,-
0049'CF 0040 8F 0000 0D53 1743 IOSB = XE_IOSB,-
0049'CF 0040 8F 0000 0D53 1744 P1 = SENSE_P1BUF,-
0049'CF 0040 8F 0000 0D53 1745 P2 = #SENSE_P2DESC
0049'CF 0040 8F 0000 0D7A 1746
0049'CF 0040 8F 0000 0D7A 1747 $CANTIM_S REQIDT = #TIMEOUT10
0049'CF 0040 8F 0000 0D89 1748
0049'CF 0040 8F 0000 0D89 1749 BSBW CHECK_IOSB ; Check status
0049'CF 0040 8F 0000 0D8C 1750
0049'CF 0040 8F 0000 0D8C 1751 MOVZWL XE_IOSB+2,R4 ; Number of bytes returned for p2 buff
0049'CF 0040 8F 0000 0D91 1752 MOVAL ADR,R5 ; Digital multicast NI address
0049'CF 0040 8F 0000 0D96 1753 MOVL #6,R7 ; Adr length
0049'CF 0040 8F 0000 0D99 1754 10$:
0049'CF 0040 8F 0000 0D99 1755 MOVAL SENSE_P2BUF,R6 ; Address of P2 buff returned
0049'CF 0040 8F 0000 0D9E 1756 MATCHC #6,(R5),R4,(R6) ; Check the parameters returned
0049'CF 0040 8F 0000 0DA3 1757 BNEQ 30$ ; Br if not match
0049'CF 0040 8F 0000 0DA5 1758 BRW ERROR_TEST ; Otherwise go to test error case
0049'CF 0040 8F 0000 0DA8 1759 30$:
0049'CF 0040 8F 0000 0DA8 1760 PUSHAL SENSE_ERRMSG ; Error message
0049'CF 0040 8F 0000 0DAC 1761 BRW FAIL_OUT ; Failure exit
0049'CF 0040 8F 0000 0DAF 1762
0049'CF 0040 8F 0000 0DAF 1763 ERROR_TEST:
0049'CF 0040 8F 0000 0DAF 1764 $SETSFMS ENBFLG = #0 ; Turn off system service mode
0049'CF 0040 8F 0000 0DB8 1765
0049'CF 0040 8F 0000 0DB8 1766 ;
0049'CF 0040 8F 0000 0DB8 1767 ; Read data with IOSM_NOW specified but no data available
0049'CF 0040 8F 0000 0DB8 1768 ;
0049'CF 0040 8F 0000 0DB8 1769 $QIOW_S - ; Read data message
0049'CF 0040 8F 0000 0DB8 1770 CHAN = XE_CHAN,-
0049'CF 0040 8F 0000 0DB8 1771 FUNC = #IOS$ READVBLK!IOSM_NOW,-
0049'CF 0040 8F 0000 0DB8 1772 IOSB = XE_IOSB,-
0049'CF 0040 8F 0000 0DB8 1773 P1 = @RECV_BUF_ADR,-
0049'CF 0040 8F 0000 0DB8 1774 P2 = #MAX_UNABOF_SIZE
0049'CF 0040 8F 0000 0DDF 1775
0000'8F 0BE6'CF B1 0DDF 1776 CMPW XE_IOSB,#SS$_ENDOFFILE ; Correct error code?
0000'8F 0BE6'CF 69 12 0DE6 1777 BNEQ ERRST_ERR ; Br if not
```



```

ODE8 1778
ODE8 1779 : Buffer not enough to hold all information from IOS_SENSEMODE
ODE8 1780 :
ODE8 1781 :
ODE8 1782 :
ODE8 1783 $QIOW_S - ; Read device characteristics
ODE8 1784 CHAN = XE_CHAN,-
ODE8 1785 FUNC = #IOS_SENSEMODE!IOSM_CTRL,-
ODE8 1786 IOSB = XE_IOSB,-
ODE8 1787 P1 = SENSE_P1BUF,-
ODE8 1788 P2 = #ERRTST_P2DESC
OE0F 1789
0000'8F OBE6'CF B1 OE0F 1790 CMPW XE_IOSB,#SS$_BUFFEROVF ; Error code = buffer overflow?
          39 12 OE16 1791 BNEQ ERRTST_ERR ; Error if not
          OE18 1792
          OE18 1793 : Bad parameter due to message too big for the buffer
          OE18 1794 :
          OE18 1795 :
          OE18 1796 $QIOW_S - ; Transmit data message with leng = 514
          OE18 1797 EFN = #XMIT_EFN,-
          OE18 1798 CHAN = XE_CHAN,-
          OE18 1799 FUNC = #IOS_WRITEVBLK,-
          OE18 1800 IOSB = XE_IOSB,-
          OE18 1801 P1 = @XMIT_BUF_ADR,-
          OE18 1802 P2 = #MAX_ONABOF_SIZE+2,-
          OE18 1803 P5 = #DIAG_BUF+HARD_ADR
          OE45 1804
          00000000'8F 50 D1 OE45 1805 CMPL R0,#SS$_IVBUFLN ; Should be invalid buffer size
                   03 12 OE4C 1806 BNEQ ERRTST_ERR
                   0076 31 OE4E 1807 BRW READ_ERRCOUNT ; Br to read and clear error count
                   OE51 1808
                   OE51 1809 ERRTST_ERR:
                   OE51 1810 PUSHAL ERRTEST_MSG ; Error message
                   0A5B'CF DF OE51 1811 BRW FAIL_OUT ; Failure exit
                   018E 31 OE55 1812
                   OE58 1813 REQUEST_FND:
                   OE58 1814 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
5C 0049'CF 06 OFFC OE5A 1815 BBS #BRDCST_NEEDEDV,FLAG,20$ ; Skip this if being CANCELED
3D 0049'CF 07 E0 OE60 1816 BBS #LONGORSHRTV,FLAG,15$ ; Skip message if short report
                   52 D4 OE66 1817 CLRL R2 ; Make an index
                   53 08F0'CF DE OE68 1818 MOVAL ADR2+6,R3 ; Set address address
                   54 081C'CF DE OE6D 1819 MOVAL ARG_LIST,R4 ; Set arglist address
                   OE72 1820 10$:
                   84 83 9A OE72 1821 MOVZBL (R3)+,(R4)+ ; Parse to bytes
                   F9 52 06 F2 OE75 1822 AOBLS #6,R2,10$
                   OE79 1823
                   OE79 1824 $FAOL_S CTRSTR = REMOTE_ID_CTR,- ; Format address
                   OE79 1825 OUTBUF = FAO_BUF,-
                   OE79 1826 OUTLEN = BUFFER_PTR,-
                   OE79 1827 PRMLST = ARG_LIST
                   OE90 1828
                   0053'CF DF OE90 1829 PUSHAL BUFFER_PTR ; Output message
                   01 DD OE94 1830 PUSHL #1
                   00741133 8F DD OE96 1831 PUSHL #UETP$TEXT+STSSK_INFO
00000000'GF 03 FB OE9C 1832 CALLS #3,G^LIB$SIGNAL
                   OEA3 1833 15$:
                   OEA3 1834 $CANTIM_S REQIDT = #TIME_ID_2 ; Kill two minute timer

```



```
081A'CF 01 DF OEAE 1835
102C'CF 01 FB OEAE 1836
0049'CF 20 A8 OEB2 1837
04 OEB7 1838
04 OEB7 1839
04 OEB7 1840 20$:
04 OEB7 1841
04 OEB7 1842
04 OEB7 1843
04 OEB7 1844
04 OEB7 1845
04 OEB7 1846
04 OEB7 1847
04 OEB7 1848
04 OEB7 1849
04 OEB7 1850
04 OEB7 1851
04 OEB7 1852
04 OEB7 1853
04 OEB7 1854
04 OEB7 1855
04 OEB7 1856
04 OEB7 1857
04 OEB7 1858
04 OEB7 1859
04 OEB7 1860
04 OEB7 1861
04 OEB7 1862
04 OEB7 1863
04 OEB7 1864
04 OEB7 1865
04 OEB7 1866
04 OEB7 1867
04 OEB7 1868
04 OEB7 1869
04 OEB7 1870
04 OEB7 1871
04 OEB7 1872
04 OEB7 1873
04 OEB7 1874
04 OEB7 1875
04 OEB7 1876
04 OEB7 1877
04 OEB7 1878
04 OEB7 1879
04 OEB7 1880
04 OEB7 1881
04 OEB7 1882
04 OEB7 1883
04 OEB7 1884
04 OEB7 1885
04 OEB7 1886
04 OEB7 1887
04 OEB7 1888
04 OEB7 1889
04 OEB7 1890
04 OEB7 1891

PUSHAL XE_CHAN1 ; Shutdown the read request channel
CALLS #1,UNA_SHUTDOWN

BISW2 #ID_FNDM,FLAG ; Set request found

RET

REQUEST_NOTFND:
.WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>

BISW2 #BRDCST_NEDEDM,FLAG ; Set the broadcast needed flag

RET

READ_ERRCOUNT:
$SETSFM_S ENBFLG = #1 ; Turn on system service mode

$SETIMR_S - ; Set ten sec timer to prevent hang
DAYTIM = TENSEC,-
ASTADR = TIME_ERR_OUT,-
REQIDT = #TIMEOUTT1

$QIOW_S - ; Read and clear the error counters
CHAN = XE_CHAN,-
FUNC = #IOS_SENSEMODE!IOSM_RD_COUNT!IOSM_CLR_COUNT!IOSM_CTRL,-
IOSB = XE_IOSB,-
P2 = #ERRCOUNT_DESC

$CANTIM_S REQIDT = #TIMEOUTT1

BSBW CHECK_IOSB ; Check IO status block
MOVZWL XE_IOSB+2,ERRCOUNT_DESC ; Set the counter size
PUSHAL ERRCOUNT_DESC ; Push address of counters
CALLS #1,COUNTER_CHECK ; and check the counters for errors
MOVL #ERRCNT_LEN,ERRCOUNT_DESC ; Reset the buffer size

CLEAN_EXIT:
BICW2 #FLAG_SHUTDNM,FLAG ; Clear the shutdown flag

$QIOW_S - ; Shut down the device
CHAN = XE_CHAN,-
FUNC = #IOS_SETMODE!IOSM_CTRL!IOSM_SHUTDOWN,-
IOSB = XE_IOSB

BSBW CHECK_IOSB ; Check IO status block

SUC_EXIT:
$STRNLOG_S LOGNAM = MODE,-
RSLLEN = BUFFER_PTR,-
RSLBUF = FAO_BUF ; Get the run mode
BICB2 #LC_BITM,BUFFER ; Convert to upper case
CMPB #^A7L/,BUFFER ; Is this a loop for ever?
BNEQ 10$ ; BR if not
BICW2 #<TEST_OVERM!ID_FNDM>,FLAG ; Reset the termination flag
INCL PASS ; Bump the pass count
$FAO_S CTRSTR = PASS_MSG,-
```



```

                                OF8F 1892      OUTLEN = BUFFER_PTR,-
                                OF8F 1893      OUTBUF = FAC_BUF,-
                                OF8F 1894      P1    = PASS,-
                                OF8F 1895      P2    = ITERATION,-
                                OF8F 1896      P3    = #0
                                DF OFAC 1897      PUSHAL BUFFER_PTR      ; Make the end of pass message
                                DD OFB0 1898      PUSHL  #1                ; Push the string desc.
                                DD OFB2 1899      PUSHL  #UETP$ TEXT!ST$K_INFO ; Push arg count
                                FB OFB8 1900      CALLS  #3,G^LIB$SIGNAL      ; Push the signal name
                                D4 OFBF 1901      CLRL   ITERATION            ; Print the end of pass message
                                31 OFC3 1902      BRW    TIME_IT              ; Reset the iteration count
                                OFC6 1903      10$:      $DASSGN_S CHAN = XE_CHAN ; Do the next pass
                                OFC6 1904
                                OFD2 1905
                                DD OFD2 1906      MOVL   #$$$ NORMAL!ST$M_INHIB_MSG,STATUS ; All done with the channel
                                OFDB 1907      $EXIT_S STATUS ; Set successful exit status
                                OFE6 1908      ; Exit with the status
                                OFE6 1909      FAIL_OUT: ; Failure exit
                                DD OFE6 1910      PUSHL  #1                ; Arg count
                                DD OFE8 1911      PUSHL  #UETP$ TEXT!ST$K_ERROR ; Signal name
                                DD OFEE 1912      MOVL   (SP),STATUS          ; Set up status
                                DD OFF3 1913      PUSHL  #3                ; Arg count
                                31 OFF5 1914      BRW    ERROR_EXIT           ; Error exit
                                OFF8 1915

```



```

OFF8 1917 .SBTTL UNA Startup Routine
OFF8 1918 :++
OFF8 1919 : FUNCTIONAL DESCRIPTION:
OFF8 1920 : This routine will be called to start up the UNA device.
OFF8 1921 : It checks IO status block and the AST parameter
OFF8 1922 :
OFF8 1923 : CALLING SEQUENCE:
OFF8 1924 : PUSHAL CHAN
OFF8 1925 : PUSHAL P2
OFF8 1926 : CALLS #2,UNA_STARTUP
OFF8 1927 :
OFF8 1928 : INPUT PARAMETERS:
OFF8 1929 : CHAN = Address to store the returned channel in
OFF8 1930 : P2 = Address of the P2 parameter list for the UNA
OFF8 1931 :
OFF8 1932 : IMPLICIT INPUTS:
OFF8 1933 : NONE
OFF8 1934 :
OFF8 1935 : OUTPUT PARAMETERS:
OFF8 1936 : CHAN = channel associated to the UNA device on startup
OFF8 1937 :
OFF8 1938 : IMPLICIT OUTPUTS:
OFF8 1939 : Error message if error
OFF8 1940 :
OFF8 1941 : COMPLETION CODES:
OFF8 1942 : IO status in XE_IOSB if error
OFF8 1943 :
OFF8 1944 : SIDE EFFECTS:
OFF8 1945 : Program exit if error
OFF8 1946 :
OFF8 1947 :--
OFF8 1948 :
OFF8 1949 UNA_STARTUP:
OFFC OFF8 1950 .WORD *M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
OFFA OFF8 1951
OFFA OFF8 1952 $QIOW_S - ; Start the controller
OFFA OFF8 1953 CHAN = @8(AP),-
OFFA OFF8 1954 FUNC = #IOS$ SETCHAR!IOSM_CTRL!IOSM_STARTUP,-
OFFA OFF8 1955 IOSB = XE_IOSB,-
OFFA OFF8 1956 ASTADR = CHK QIO_AST,-
OFFA OFF8 1957 ASTPRM = #PRM,-
OFFA OFF8 1958 P2 = 4(AP),-
OFFA OFF8 1959 P3 = #RECVPOOL_SIZ,-
OFFA OFF8 1960 P6 = #DIAG_BUF-
04 102B 1961 RET

```



```

102C 1963      .SBTTL  UNA Shutdown Routine
102C 1964      :++
102C 1965      : FUNCTIONAL DESCRIPTION:
102C 1966      :   This routine will be called to shutdown the UNA device.
102C 1967      :   It checks IO status block and the AST parameter
102C 1968      :
102C 1969      : CALLING SEQUENCE:
102C 1970      :   PUSHAL CHAN
102C 1971      :   CALLS  #1,UNA_SHUTDOWN
102C 1972      :
102C 1973      : INPUT PARAMETERS:
102C 1974      :   CHAN = Address to store the returned channel in
102C 1975      :
102C 1976      : IMPLICIT INPUTS:
102C 1977      :   NONE
102C 1978      :
102C 1979      : OUTPUT PARAMETERS:
102C 1980      :   NONE
102C 1981      :
102C 1982      : IMPLICIT OUTPUTS:
102C 1983      :   Error message if error
102C 1984      :
102C 1985      : COMPLETION CODES:
102C 1986      :   IO status in XE_IOSB if error
102C 1987      :
102C 1988      : SIDE EFFECTS:
102C 1989      :   Program exit if error
102C 1990      :
102C 1991      :--
102C 1992      :
102C 1993      : UNA_SHUTDOWN:
OFFC 102C 1994      : .WORD  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
102E 1995      :
102E 1996      : $QIOW_S - ; Shut down the device
102E 1997      :   CHAN = @4(AP),-
102E 1998      :   FUNC = #IOS$ SETMODE!IOSM_CTRL!IOSM_SHUTDOWN,-
102E 1999      :   IOSB = XE_IOSB,-
102E 2000      :   ASTADR = CHK_QIO_AST,-
102E 2001      :   ASTPRM = #PRM
04  1056 2002      RET

```



```

1057 2004 .SBTTL CHECKIOSB - Check IO status block
1057 2005 :++
1057 2006 : FUNCTIONAL DESCRIPTION:
1057 2007 : This routine checks the IO status block = #SS$_NORMAL
1057 2008 :
1057 2009 : CALLING SEQUENCE:
1057 2010 : BSBW CHECK_IOSB
1057 2011 :
1057 2012 : INPUT PARAMETERS:
1057 2013 : NONE
1057 2014 :
1057 2015 : IMPLICIT INPUTS:
1057 2016 : NONE
1057 2017 :
1057 2018 : OUTPUT PARAMETERS:
1057 2019 : NONE
1057 2020 :
1057 2021 : IMPLICIT OUTPUTS:
1057 2022 : Exit with status if IOSB not right
1057 2023 :
1057 2024 : COMPLETION CODES:
1057 2025 : IO status in STATUS if error
1057 2026 :
1057 2027 : SIDE EFFECTS:
1057 2028 : Program exit if error found
1057 2029 :
1057 2030 :--
1057 2031 CHECK_IOSB:
0000'8F OBE6'CF B1 1057 2032 CMPW XE IOSB,#SS$_NORMAL ; Is the QIO O.K.?
01 12 105E 2033 BNEQ 10$ ; Br if not
05 1060 2034 RSB ; Return
1061 2035 10$:
7E OBE6'CF 3C 1061 2036 MOVZWL XE IOSB,-(SP) ; Push the error status code
0769'CF 6E D0 1066 2037 MOVL (SP),STATUS ; Set return status
01 DD 106B 2038 PUSHL #1 ; Argument count
050F 31 106D 2039 BRW ERROR_EXIT ; Error exit
1070 2040

```



```

1070 2042 .SBTTL Check QIO AST Routine
1070 2043 :++
1070 2044 : FUNCTIONAL DESCRIPTION:
1070 2045 : This routine will be called as a QIO completion AST routine
1070 2046 : It checks IO status block and the AST parameter
1070 2047 :
1070 2048 : CALLING SEQUENCE:
1070 2049 : Called via AST at $QIO completion
1070 2050 :
1070 2051 : INPUT PARAMETERS:
1070 2052 : NONE
1070 2053 :
1070 2054 : IMPLICIT INPUTS:
1070 2055 : NONE
1070 2056 :
1070 2057 : OUTPUT PARAMETERS:
1070 2058 : NONE
1070 2059 :
1070 2060 : IMPLICIT OUTPUTS:
1070 2061 : Error message if error
1070 2062 :
1070 2063 : COMPLETION CODES:
1070 2064 : IO status in STATUS if error
1070 2065 :
1070 2066 : SIDE EFFECTS:
1070 2067 : Program exit if error
1070 2068 :
1070 2069 :--
1070 2070 CHK_QIO_AST:
1070 2071 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
1070 2072 BSBW CHECK_IOSB ; Go check IO status block
1070 2073 CMPL #PRM,4(AP) ; Check AST parameter
1070 2074 BNEQ 10$ ; Branch if not #1 (STARTUP)
1070 2075 RET
1070 2076 10$:
1070 2077 PUSHAL ASTPAR_ERRMSG ; Error message
1070 2078 BRW FAIL_OUT

```

04 AC 00000064 8F 01 0A77'CF DF FF5F 31


```

1087 2080 .SBTTL Remote Loopback I/O Timeout AST Routine
1087 2081 :++
1087 2082 : FUNCTIONAL DESCRIPTION:
1087 2083 : This routine services the completion AST if during the remote loopback
1087 2084 : test we time out while waiting for a message to be returned. It forces
1087 2085 : the test to continue by setting the event flag for which we were waiting
1087 2086 : when we timed out.
1087 2087 :
1087 2088 : CALLING SEQUENCE:
1087 2089 : Called via AST
1087 2090 :
1087 2091 : INPUT PARAMETERS:
1087 2092 : 04(AP) - Ignored
1087 2093 :
1087 2094 : IMPLICIT INPUTS:
1087 2095 : NONE
1087 2096 :
1087 2097 : OUTPUT PARAMETERS:
1087 2098 : NONE
1087 2099 :
1087 2100 : IMPLICIT OUTPUTS:
1087 2101 : NONE
1087 2102 :
1087 2103 : COMPLETION CODES:
1087 2104 : Status from $SETEF
1087 2105 :
1087 2106 : SIDE EFFECTS:
1087 2107 : Another timer AST is enabled.
1087 2108 : Synchronous test continues because event flag is set.
1087 2109 :
1087 2110 :--
1087 2111 :
1087 2112 LOOPBACK_TMO:
1087 2113 .WORD ^M<>
1089 2114
1089 2115 BLBC FLAG,5$ ; Skip this msg if normal running...
108E 2116 ; ...since there will be another anyway
108E 2117 $FAO_S CTRSTR = LOOPBACK_TMO_MSG,-
108E 2118 OUTLEN = BUFFER_PTR,-
108E 2119 OUTBUF = FAO_BUF,-
108E 2120 P1 = #0 =
108E 2121 P2 = ITERATION
10A7 2122 BSBW DUMP SIGNAL
10AA 2123 CALLS #0,DUMP STAMPS
10AF 2124 5$: MOVAL -(SP),R0 ; Create STATE holder for following
10B2 2125 $READEF_S EFN = #XMIT_EFN,- ; Check to see...
10B2 2126 STATE = (R0) ; ...if we hung while trying to transmit
10BD 2127 BITL #XMIT_EFN,(SP)+ ; Have we transmitted successfully?
10C0 2128 ; Note: flag is set except while in...
10C0 2129 ; ...the $QIOW in REMOTE_LOOPBACK_TEST
10C0 2130 BNEQ 10$ ; BR if transmit is complete - we're...
10C2 2131 ; ...waiting for recv, and that will
10C2 2132 ; ...eventually terminate
10C2 2133 BBC #TEST_OVERV,FLAG,10$ ; BR if overall timer hasn't rung yet
10C8 2134 PUSHAL TIMEOUT9 ; We seem to be hung in transmit. Abort
10CC 2135 CALLS #1,TIME_ERR_OUT
10D1 2136 10$:

```


UETUNAS00
V04-000

D 11
VAX/VMS UETP DEVICE TEST FOR THE UNA 16-SEP-1984 01:37:49 VAX/VMS Macro V04-00 Page 50
Remote Loopback I/O Timeout AST Routine 5-SEP-1984 04:26:50 [UETP.SRC]UETUNAS00.MAR;1 (20)

10D1	2137	\$SETIMR_S DAYTIM = HALFMIN,-	; Allow time for messages to complete
10D1	2138	ASTADR = LOOPBACK_TMO,-	
10D1	2139	REQIDT = #LOOPBACK_TMO	
10E7	2140	\$SETEF_S EFN = #RECV_EFN	
04	10F0	2141	RET


```

10F1 2143      .SBTTL Remote Loopback I/O Completion AST Routine
10F1 2144      :++
10F1 2145      : FUNCTIONAL DESCRIPTION:
10F1 2146      :   This routine services the completion AST from the remote loopback test's
10F1 2147      :   attempt to read. It has to be a bit more forgiving than most routines,
10F1 2148      :   since we may have $CANCELled the I/O due to a timeout. The timeout
10F1 2149      :   happens occasionally since not all Ethernet messages are delivered.
10F1 2150      :
10F1 2151      : CALLING SEQUENCE:
10F1 2152      :   Called via AST at $QIO READ
10F1 2153      :
10F1 2154      : INPUT PARAMETERS:
10F1 2155      :   04(AP) - Index into SIZE_TBL1 for the size of the message we attempted
10F1 2156      :   to read. This index may be used for LOOPBACK_ACK as well, a
10F1 2157      :   parallel table.
10F1 2158      :
10F1 2159      : IMPLICIT INPUTS:
10F1 2160      :   RCV_IOSB - $QIO read completion status
10F1 2161      :   SIZE_TBL1 - Table of message sizes
10F1 2162      :   ITERATION - Count of successful messages sent
10F1 2163      :
10F1 2164      : OUTPUT PARAMETERS:
10F1 2165      :   ITERATION - Incremented if this message was successful
10F1 2166      :
10F1 2167      : IMPLICIT OUTPUTS:
10F1 2168      :   LOOPBACK_ACK - Bit set to indicate that a particular message size was
10F1 2169      :   received
10F1 2170      :
10F1 2171      : COMPLETION CODES:
10F1 2172      :   Status from RECV_AST
10F1 2173      :
10F1 2174      : SIDE EFFECTS:
10F1 2175      :   Message and program exit if error found
10F1 2176      :
10F1 2177      :--
10F1 2178
10F1 2179      LOOPBACK_AST:
10F1 2180      .WORD      ^M<R6,R7>
10F3 2181
10F3 2182      MOVL      04(AP),R6                ; Get index into SIZE_TBL1
10F7 2183      MOVL      RECV_BUF_ADR,R7      ; Get pointer to message buffer
10FC 2184      PUSHL      10(R7)
10FF 2185      MOVZWL     RCV_IOSB+2,-(SP)
1104 2186      CALLS      #2,RECV_STAMP      ; Remember message received if dump mode
1109 2187      CMPW      #$$$_ABORT,RCV_IOSB ; Did we $CANCEL this transmission?
1110 2188      BEQL      20$                ; BR if so, no point in normal checks
1112 2189      BLBC      FLAG,10$            ; Skip sequence check if normal running
1117 2190      CMPL      ITERATION,10(R7)    ; Check message sequence number...
111D 2191      BEQL      10$                ; ...and skip if it's OK
111F 2192      $FAO_S      CTRSTR = OUT_OF_SEQ_MSG,-
111F 2193      OUTLEN = BUFFER_PTR,-
111F 2194      OUTBUF = FAO_BUF,-
111F 2195      P1 = ITERATION,-
111F 2196      P2 = SIZE_TBL1[R6],-
111F 2197      P3 = 10(R7),-
111F 2198      P4 = RCV_IOSB+2,-
111F 2199      P5 = #0

```

```

00C0
56 04 AC D0
57 0BFC'CF D0
   0A A7 DD
7E 0BFO'CF 3C
1221'CF 02 FB
OBEE'CF 0000'8F B1
   50 13
   3B 0049'CF E9
OA A7 0791'CF D1
   33 13

```


1256'CF	0176	30	1144	2200	BSBW	DUMP SIGNAL	
OA A7	0791'CF	FB	1147	2201	CALLS	#0,DUMP_STAMPS	; List the last several msgs sent & recv
		D0	114C	2202	MOVL	ITERATION,10(R7)	; Fix up sequence number...
			1152	2203			; ...so data comparison can be done
			1152	2204			
0C54'C6	01	88	1152	2205	BISB2	#1,LOOPBACK_ACK(R6)	; Flag that this size was received
099E'CF	46	DD	1157	2206	PUSHL	SIZE_TBL1[R6]	; Now call general routine...
1182'CF	01	FB	115C	2207	CALLS	#1,RCV_AST	; ...to check message's correctness
		04	1161	2208	RET		
			1162	2209			
			1162	2210	\$FAO_S	CTRSTR = ABORT MSG,-	
			1162	2211		OUTLEN = BUFFER_PTR,-	
			1162	2212		OUTBUF = FAO_BUF,-	
			1162	2213		P1 = ITERATION	
1256'CF	0141	30	1179	2214	BSBW	DUMP SIGNAL	
	00	FB	117C	2215	CALLS	#0,DUMP_STAMPS	; List the last several msgs sent & recv
		04	1181	2216	RET		


```
1182 2218 .SBTTL Receive data AST routine
1182 2219 :++
1182 2220 : FUNCTIONAL DESCRIPTION:
1182 2221 : This routine will be called as receive data AST routine
1182 2222 : It checks IO status and compare the data in the receive buffer
1182 2223 : against the transmit buffer
1182 2224 :
1182 2225 : CALLING SEQUENCE:
1182 2226 : Called via AST at $QIO READ
1182 2227 :
1182 2228 : INPUT PARAMETERS:
1182 2229 : AST parameter = message length
1182 2230 :
1182 2231 : IMPLICIT INPUTS:
1182 2232 : NONE
1182 2233 :
1182 2234 : OUTPUT PARAMETERS:
1182 2235 : NONE
1182 2236 :
1182 2237 : IMPLICIT OUTPUTS:
1182 2238 : Error message if error found
1182 2239 :
1182 2240 : COMPLETION CODES:
1182 2241 : in STATUS
1182 2242 :
1182 2243 : SIDE EFFECTS:
1182 2244 : Program exit if error found
1182 2245 :
1182 2246 :--
1182 2247 RECV_AST:
1182 2248 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
1184 2249
1184 2250 MOVL 4(AP),R4 ; Message length as ast param
1188 2251 MOVL RECV_BUF_ADR,R5 ; Address of receive buffer
118D 2252 MOVL XMIT_BUF_ADR,R6 ; Address of transmit buffer
1192 2253 CMPW (R5),#8 ; Is this an external transfer with a...
1195 2254 ; ...skip count modification?
1195 2255 BNEQ 10$ ; BR if not
1197 2256 CLRW (R5) ; Reset the skip count to zero
1199 2257 10$:
1199 2258 CMPW #SS$_DATAOVERUN,RCV_IOSB ; Is our buffer too small for message?
11A0 2259 BEQL 12$ ; Length problems are recoverable (below)
11A2 2260 CMPW #SS$_NORMAL,RCV_IOSB ; Is the QIO O.K.?
11A9 2261 BNEQ 20$ ; BR if not
11AB 2262 CMPW RCV_IOSB+2,R4 ; Did we get an entire message?
11B0 2263 BEQL 15$ ; BR if we did
11B2 2264 $FAO_S CTRSTR = BADLEN,- ; Complain if we did not...
11B2 2265 OUTLEN = BUFFER_PTR,-
11B2 2266 OUTBUF = FAO_BUF,-
11B2 2267 P1 = ITERATION,-
11B2 2268 P2 = RCV_IOSB+2,- ; ...telling actual size...
11B2 2269 P3 = R4 ; ...versus requested size
11CF 2270 BSBW DUMP_SIGNAL
11D2 2271 CALLS #0,DUMP_STAMPS ; If in DUMP mode, print a history
11D7 2272 RET ; Treat this as a recoverable error
11D8 2273 15$:
11D8 2274 CMPC3 R4,(R5),(R6) ; Compare the data
```



```

10      12 11DC 2275      BNEQ 30$
        04 11DE 2276      RET
        11DF 2277 20$:
7E      0BEE'CF 3C 11DF 2278      MOVZWL RCV_IOSB,-(SP)      ; Push the error status code
0769'CF 6E  D0 11E4 2279      MOVL  (SPT,STATUS)      ; Set return status
        01  DD 11E9 2280      PUSHL  #1      ; Argument count for the error message
        0391 31 11EB 2281      BRW  ERROR_EXIT      ; Error exit
        11EE 2282
        11EE 2283 30$:
        7E 61 9A 11EE 2284      MOVZBL (R1),-(SP)      ; Get good byte
        7E 63 9A 11F1 2285      MOVZBL (R3),-(SP)      ; Get bad byte
7E      54 50 C3 11F4 2286      SUBL3  R0,R4,-(SP)      ; Get the byte number
5B      07B8'CF 000007B8'8F C1 11F8 2287      ADDL3  #UNIT_LIST,UNIT_LIST,R11 ; Get the unit block address
        14 AB DF 1202 2288      PUSHAL UETUNT$T_FILSPC(R11) ; Use it to point to the device name
        04 DD 1205 2289      PUSHL  #4      ; Push arg count
        0074801A 8F DD 1207 2290      PUSHL  #UETP$_DATADEVERR!ST$K_ERROR ; Push the signal name
        06 DD 120D 2291      PUSHL  #6      ; Push arg count
        0074801A 8F D0 120F 2292      MOVL  #UETP$_DATADEVERR!ST$K_ERROR,- ; Push the signal name
        0769'CF 1215 2293      STATUS
        0364 31 1218 2294      BRW  ERROR_EXIT      ; Failure exit
```



```

121B 2296 .SBTTL DUMP Mode Routines
121B 2297 ;+
121B 2298 ; Being a set of routines to keep track of the last several messages sent
121B 2299 ; and received during the remote loopback test if running in DUMP mode.
121B 2300 ; To call XMIT_STAMP ($QIO transmit time stamp) or RECV_STAMP (AST received
121B 2301 ; time stamp):
121B 2302 ; PUSHL MESSAGE-SEQUENCE-NUMBER
121B 2303 ; PUSHL BUFFER-LENGTH
121B 2304 ; CALLS #1,{XMIT,RECV}_STAMP
121B 2305 ;
121B 2306 ; To call DUMP_STAMPS (dump recent time stamps):
121B 2307 ; CALLS #0,DUMP_STAMPS
121B 2308 ;
121B 2309 .ENABLE LSB
121B 2310 XMIT_STAMP:
121B 2311 .WORD ^M<R2>
121D 2312 CLRL R0 ; Remember that we're XMIT_STAMP
121F 2313 BRB 10$
1221 2314 RECV_STAMP:
1221 2315 .WORD ^M<R2>
1223 2316 MOVL #1,R0 ; Remember that we're RECV_STAMP
1226 2317 10$: MOVL SBP,R2 ; Get pointer into time stamp list
122B 2318 MOVW R0,(R2)+ ; Save RECV/XMIT...
122E 2319 MOVW 04(AP),(R2)+ ; ...length of message...
1232 2320 MOVL 08(AP),(R2)+ ; ...message sequence number...
1236 2321 $GETTIM,S TIMADR = (R2) ; ...and time when sent or received
123F 2322 ADDL2 #8,R2 ; Point to the next time stamp slot
1242 2323 CMPL #STAMP_BUFFER+TSCLL,R2 ; It's a circular list - have we...
1249 2324 BGTR 20$ ; ...gone around? BR if we have not
124B 2325 MOVAL STAMP_BUFFER,R2 ; Reuse slots at beginning if we have
1250 2326 20$: MOVL R2,SBP ; Save pointer into time stamp list
1255 2327 RET
1256 2328 .DISABLE LSB
1256 2329
1256 2330 DUMP_STAMPS:
1256 2331 .WORD ^M<R2>
1258 2332 BLBC FLAG,30$ ; Skip dump if not in DUMP mode
125D 2333 $FAO_S CTRSTR = TIME_STAMP_INTRO_MSG,-
125D 2334 OUTLEN = BUFFER_PTR,-
125D 2335 OUTBUF = FAO_BUF,-
125D 2336 P1 = RCV_IOSB,-
125D 2337 P2 = RCV_IOSB+4
1278 2338 BSBB DUMP_SIGNAL ; Indicate what's to follow
127A 2339 MOVL SBP,R2 ; Pick up pointer into time stamp list
127F 2340 10$: SUBL2 #16,R2 ; Point to previous slot
1282 2341 CMPL #STAMP_BUFFER-16,R2 ; Are we at the start of the list?
1289 2342 BLSS 20$ ; BR if not - slot is legit
128B 2343 MOVAL STAMP_BUFFER+TSCLL-16,R2 ; Set pointer to last slot
1290 2344 20$: CMPL R2,SBP ; Have we gone full circle yet?
1295 2345 BEQL 30$ ; BR if we have - exit routine
1297 2346 MOVAL 8(R2),R0 ; Get address of time stamp in slot
1298 2347 $FAO_S CTRSTR = TIME_STAMP_MSG,-
1298 2348 OUTLEN = BUFFER_PTR,-
1298 2349 OUTBUF = FAO_BUF,-
1298 2350 P1 = (R2),-
1298 2351 P2 = 2(R2),-
1298 2352 P3 = 4(R2),-

```


UETUNAS00
V04-000

VAX/VMS UETP DEVICE TEST FOR THE UNA
DUMP Mode Routines

J 11

16-SEP-1984 01:37:49 VAX/VMS Macro V04-00
5-SEP-1984 04:26:50 [UETP.SRC]UETUNAS00.MAR;1

Page 56
(23)

```
03 10 129B 2353 P4 = R0
C3 11 12B8 2354 DUMP_SIGNAL
04 12BA 2355 10$ ; Type one time stamp...
12BC 2356 30$: ; ...and loop for the next
12BD 2357
12BD 2358 DUMP_SIGNAL:
DF 12BD 2359 PUSHAL BUFFER_PTR
01 DD 12C1 2360 PUSHL #1
00741133 8F DD 12C3 2361 PUSHL #UETP$ TEXT:ST$K_INFO
00000000'GF 03 FB 12C9 2362 CALLS #3,G^LIB$SIGNAL
05 12D0 2363 RSB
```



```

12D1 2365 .SBTTL COUNTER_CHECK Routine
12D1 2366 :++
12D1 2367 : FUNCTIONAL DESCRIPTION:
12D1 2368 : This routine is the UNA counters checking routine. A list of
12D1 2369 : specific UNA counters which are considered to indicate errors
12D1 2370 : is referenced at CNTR_TBL. When the counters are read their
12D1 2371 : value is checked to see if it is non zero. If non zero a warning
12D1 2372 : message will be printed containing the name of the counter and
12D1 2373 : it's value.
12D1 2374 :
12D1 2375 : CALLING SEQUENCE:
12D1 2376 : PUSHAL COUNTER_DESC
12D1 2377 : CALLS #1, COUNTER_CHECK
12D1 2378 :
12D1 2379 : INPUT PARAMETERS:
12D1 2380 : 4(AP) = Address of the counters to check.
12D1 2381 :
12D1 2382 : IMPLICIT INPUTS:
12D1 2383 : NONE
12D1 2384 :
12D1 2385 : OUTPUT PARAMETERS:
12D1 2386 : Sets the error occurred flag if a warning message is printed.
12D1 2387 :
12D1 2388 : IMPLICIT OUTPUTS:
12D1 2389 : Error or success messages
12D1 2390 :
12D1 2391 : COMPLETION CODES:
12D1 2392 : NONE
12D1 2393 :
12D1 2394 : SIDE EFFECTS:
12D1 2395 : NONE
12D1 2396 :
12D1 2397 :--
12D1 2398
12D1 2399 COUNTER_CHECK:
12D1 2400 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
12D3 2401
12D3 2402
12D7 2403
12DA 2404
12E1 2405
12E5 2406
12EA 2407
12EA 2408
12EF 2409
12F1 2410
12F4 2411
12F4 2412
12F7 2413
12F9 2414
12FA 2415
12FC 2416
12FC 2417
12FF 2418
1301 2419
1302 2420
1304 2421

OBE2'CF 56 04 AC D0 12D3 2401
57 66 3C 12D7 2403
04 A6 57 C1 12DA 2404
56 04 A6 D0 12E1 2405
59 088D'CF DE 12E5 2406
OBE2'CF 56 D1 12EA 2407
03 12 12EF 2408
00DC 31 12F1 2409
58 86 3C 12F4 2411
00 EF 12F4 2412
OC 12F7 2413
58 58 12F9 2414
12FA 2415
5A 89 3C 12FC 2416
00 EF 12FC 2417
OC 1301 2419
5B 5A 1302 2420
5B 58 D1 1304 2421

CHK_LOOP:
CMPL R6,END_ADR ; All done?
BNEQ 10$ ; BR if not...
BRW COUNTER_EXIT ; ...else bail out

10$:
MOVZWL (R6)+,R8 ; Get cntrl desc
EXTZV #NMA$V_CNT_TYP,-
#NMA$S_CNT_TYP,-
R8,R8 ; Get the counter type

20$:
MOVZWL (R9)+,R10 ; Get a table code
EXTZV #NMA$V_CNT_TYP,-
#NMA$S_CNT_TYP,-
R10,R11 ; Get the counter type
CMPL R8,R11 ; Is this it?

```



```

      4B 13 1307 2422 BEQL 80$ ; BR if yes
      59 04 C0 1309 2423 ADDL2 #4,R9 ; Skip name pointer
000008B7'8F 59 D1 130C 2424 CMPL R9,#TBL_END ; End of table?
      E7 12 1313 2425 BNEQ 20$ ; BR if not
      59 088D'CF DE 1315 2426 MOVAL CNTR TBL,R9 ; Set table address
      OD EF 131A 2427 EXTZV #NMASV_CNT_WID,-
      02 131C 2428 #NMASV_CNT_WID,-
      58 FE A6 131D 2429 -2(R6),R8 ; Get the counter width
03 FE A6 OC E1 1320 2430 BBC #NMASV_CNT_MAP,-2(R6),30$ ; If not a mapped counter then carry on el
      56 02 C0 1325 2431 ADDL2 #2,R6 ; ...skip the map word
      02 01 58 8F 1328 2432 30$: CASEB R8,#1,#2 ; Skip the counter
      132C 2433 40$:
      0017' 132C 2435 .WORD 50$-40$
      001C' 132E 2436 .WORD 60$-40$
      0022' 1330 2437 .WORD 70$-40$
      029A'CF DF 1332 2438 PUSHAL CASE_FAILED ; Push the string address
      01 DD 1336 2439 PUSHL #1 ; Push the argument count
00741132 8F DD 1338 2440 PUSHL #UETPS_TEXT!STSSK_ERROR ; Push the signal name
      03 DD 133E 2441 PUSHL #3 ; Push the argument count
      023C 31 1340 2442 BRW ERROR_EXIT ; Thats it
      1343 2443 50$: INCL R6 ; Skip a byte counter
      56 D6 1343 2444 BRW CHK_LOOP
      FFA2 31 1345 2445 60$: ADDL2 #2,R6 ; Skip a word counter
      56 02 C0 1348 2446 BRW CHK_LOOP
      FF9C 31 134B 2447 70$: ADDL2 #4,R6 ; Skip a long word counter
      56 04 C0 134E 2448 BRW CHK_LOOP
      FF96 31 1351 2449 80$: EXTZV #NMASV_CNT_WID,-
      OD EF 1354 2452 #NMASV_CNT_WID,-
      02 1356 2453 -2(R6),R8 ; Get the counter width
      58 FE A6 1357 2454 BBC #NMASV_CNT_MAP,-2(R6),90$ ; If not a mapped counter then carry on el
03 FE A6 OC E1 135A 2455 ADDL2 #2,R6 ; ...skip the map word
      56 02 C0 135F 2456 90$: CASEB R8,#1,#2 ; Skip the counter
      02 01 58 8F 1362 2457 100$:
      1366 2460 .WORD 110$-100$
      0017' 1366 2461 .WORD 120$-100$
      001F' 1368 2462 .WORD 130$-100$
      0027' 136A 2463 PUSHAL CASE_FAILED ; Push the string address
      029A'CF DF 136C 2464 PUSHL #1 ; Push the argument count
      01 DD 1370 2465 PUSHL #UETPS_TEXT!STSSK_ERROR ; Push the signal name
00741132 8F DD 1372 2466 PUSHL #3 ; Push the argument count
      03 DD 1378 2467 BRW ERROR_EXIT ; Thats it
      0202 31 137A 2468 110$: MOVZBL (R6)+,COUNTER ; Get a byte counter
      OBDE'CF 86 9A 137D 2470 BRW 140$
      000D 31 1382 2471 120$: MOVZWL (R6)+,COUNTER ; Get a word counter
      OBDE'CF 86 3C 1385 2473 BRW 140$
      0005 31 138A 2474 130$: MOVL (R6)+,COUNTER ; Get a long word counter
      OBDE'CF 86 D0 138D 2475 140$:
      138D 2476 BNEQ 150$
      08 12 1392 2478
```



```
59 088D'CF DE 1394 2479 MOVAL CNTR_TBL,R9 ; Reset the table address
    FF4E 31 1399 2480 BRW CHK_LOOP ; BR if counter was zero
    139C 2481 150$: $FAO_S CTRSTR = COUNTER_MSG,- ; Generate a bad counter message
    139C 2482 OUTLEN = BUFFER_PTR,-
    139C 2483 OUTBUF = FAO_BUF,-
    139C 2484 P1 = (R9),-
    139C 2485 P2 = COUNTER
    139C 2486
    0053'CF DF 13B5 2487 PUSHAL BUFFER_PTR ; Push the string address
    01 DD 13B9 2488 PUSHL #1 ; Push the argument count
    00741133 8F DD 13BB 2489 PUSHL #UETP$ TEXT!ST$SK_INFO ; Push the signal name
00000000'GF 03 FB 13C1 2490 CALLS #3, G^CIB$SIGNAL ; Print the error message
59 088D'CF DE 13C8 2491 MOVAL CNTR_TBL,R9 ; Reset the counter table pointer
    FF1A 31 13CD 2492 BRW CHK_LOOP ; Thats it
    13D0 2493
    13D0 2494 COUNTER_EXIT:
04 13D0 2495 RET
```



```

13D1 2497 .SBTTL Half Minute Timer Expiration Routine
13D1 2498 :++
13D1 2499 : FUNCTIONAL DESCRIPTION:
13D1 2500 : This routine will be called only if the timer which was set to prevent
13D1 2501 : program hangs goes off.
13D1 2502 :
13D1 2503 : CALLING SEQUENCE:
13D1 2504 : Called via AST at $SETIMR expiration.
13D1 2505 :
13D1 2506 : INPUT PARAMETERS:
13D1 2507 : NONE
13D1 2508 :
13D1 2509 : IMPLICIT INPUTS:
13D1 2510 : NONE
13D1 2511 :
13D1 2512 : OUTPUT PARAMETERS:
13D1 2513 : NONE
13D1 2514 :
13D1 2515 : IMPLICIT OUTPUTS:
13D1 2516 : Time out error message
13D1 2517 :
13D1 2518 : COMPLETION CODES:
13D1 2519 : #SS$_TIMEOUT
13D1 2520 :
13D1 2521 : SIDE EFFECTS:
13D1 2522 : Program exit
13D1 2523 :
13D1 2524 : --
13D1 2525 :
13D1 2526 TIME_ERR_OUT:
13D1 2527 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
00000000'8F DD 13D3 2528 PUSHL #SS$_TIMEOUT ; Push the signal name = timeout
0769'CF 6E DO 13D9 2529 MOVL (SP),STATUS ; Set exit status
04 AC DD 13DE 2530 PUSHL 4(AP) ; Push the REQIDT which is the timeout
13E1 2531 ; string address
01 DD 13E1 2532 PUSHL #1 ; Push the argument count total
00741132 8F DD 13E3 2533 PUSHL #UETP$_TEXT!STSSK_ERROR ; Push the message number
04 DD 13E9 2534 PUSHL #4 ; Push the argument count
0191 31 13EB 2535 BRW ERROR_EXIT ; Bail out completely

```


0049'CF 02 OFFC AB 04


```

13F6 2573 .SBTTL System Service Exception Handler
13F6 2574 :++
13F6 2575 : FUNCTIONAL DESCRIPTION:
13F6 2576 : This routine is executed if a software or hardware exception occurs or
13F6 2577 : if a LIB$SIGNAL system service is used to output a message.
13F6 2578 : Information about this method of handling messages and errors can be
13F6 2579 : found in the VMS COMMON RUN-TIME manual and in the VMS SYSTEM SERVICE
13F6 2580 : manual.
13F6 2581 :
13F6 2582 : CALLING SEQUENCE:
13F6 2583 : Entered via an exception from the system
13F6 2584 :
13F6 2585 : INPUT PARAMETERS:
13F6 2586 : ERROR_COUNT = previous cumulative error count
13F6 2587 :
13F6 2588 : AP ---->
13F6 2589 :
13F6 2590 :
13F6 2591 :
13F6 2592 :
13F6 2593 :
13F6 2594 :
13F6 2595 :
13F6 2596 :
13F6 2597 :
13F6 2598 :
13F6 2599 :
13F6 2600 :
13F6 2601 :
13F6 2602 :
13F6 2603 :
13F6 2604 :
13F6 2605 :
13F6 2606 :
13F6 2607 :
13F6 2608 :
13F6 2609 :
13F6 2610 :
13F6 2611 :
13F6 2612 :
13F6 2613 :
13F6 2614 :
13F6 2615 : IMPLICIT INPUTS:
13F6 2616 : NONE
13F6 2617 :
13F6 2618 : OUTPUT PARAMETERS:
13F6 2619 : NONE
13F6 2620 :
13F6 2621 : IMPLICIT OUTPUTS:
13F6 2622 : NONE
13F6 2623 :
13F6 2624 : COMPLETION CODES:
13F6 2625 : NONE
13F6 2626 :
13F6 2627 : SIDE EFFECTS:
13F6 2628 : May branch to ERROR_EXIT
13F6 2629 :--

```

2	
SIGNL ARY PNT	
MECH ARY PNT	
4	
ESTABLISH FP	
DEPTH	Mechanism Array
R0	
R1	
N	
CONDITION NAME	
N-3 ADDITIONAL LONG WORD ARGS	Signal Array
PC	
PSL	


```

13F6 2630
13F6 2631 SSERROR:
OFFC 13F6 2632 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
13F8 2633
13F8 2634 $SETAST_S ENBFLG = #0 ; Disable AST delivery
50 01 DD 1401 2635 PUSHL #1 ; Assume ASTs were enabled
00' D1 1403 2636 CMPL S^#SS$_WASSET,R0 ; Were ASTs enabled?
02 13 1406 2637 BEQL 10$ ; BR if they were
6E D4 1408 2638 CLRL (SP) ; Set ASTs to remain disabled
140A 2639 10$:
140A 2640 $SETSFM_S ENBFLG = #0 ; Disable SS failure mode
50 01 DD 1413 2641 PUSHL #1 ; Assume SS failure mode was enabled
00' D1 1415 2642 CMPL S^#SS$_WASSET,R0 ; Was SS failure mode enabled?
02 13 1418 2643 BEQL 20$ ; BR if it was
6E D4 141A 2644 CLRL (SP) ; Set SS failure mode to remain off
141C 2645 20$:
56 04 AC D0 141C 2646 MOVL CHF$_SIGARGLST(AP),R6 ; Get the signal array pointer
59 04 A6 7D 1420 2647 MOVQ CHF$_SIG_NAME(R6),R9 ; Get NAME in R9 and ARG1 in R10
10 ED 1424 2648 CMPZV #ST$_FAC_NO,- ; Is this a message from LIB$SIGNAL?
0C 1426 2649 #ST$_FAC_NO,-
00000074 8F 59 1427 2650 R9,#UETP$_FACILITY
14 12 142D 2651 BNEQ 30$ ; BR if this is not a UETP exception
66 02 C2 142F 2652 SUBL2 #2,CHF$_SIG_ARGS(R6) ; Drop the PC and PSL
21 11 1432 2653 $PUTMSG_S MSGVEC= CHF$_SIG_ARGS(R6) ; Print the message
1441 2654 BRB 40$ ; Restore ASTs and SS fail mode
1443 2655 30$:
59 00000000' 8F D1 1443 2656 CMPL #SS$_SSFAIL,R9 ; RMS failures are SysSvc failures
32 12 144A 2657 BNEQ 50$ ; BR if this can't be an RMS failure
10 ED 144C 2658 CMPZV #ST$_FAC_NO,- ; Is it an RMS failure?
0C 144E 2659 #ST$_FAC_NO,-
01 5A 144F 2660 R10,#RMS$_FACILITY
2B 12 1451 2661 BNEQ 50$ ; BR if not
5A F0000000 8F CA 1453 2662 BICL2 #^XF0000000,R10 ; Strip control bits from status code
08 A6 04 39 145A 2663 MATCHC #4,CHF$_SIG_ARG1(R6),- ; Is it an RMS failure for which...
14 145E 2664 #NRAT_LENGTH,-
005E' CF 145F 2665 NO_RMS_AST_TABLE ; ...no AST can be delivered?
1A 13 1462 2666 BEQL 50$ ; BR if so - must give error here
1464 2667 40$:
01 BA 1464 2668 POPR #^M<R0> ; Restore SS failure mode...
1466 2669 $SETSFM_S ENBFLG = R0 ;
01 BA 146F 2670 POPR #^M<R0> ; Restore AST enable...
1471 2671 $SETAST_S ENBFLG = R0 ;
50 00' D0 147A 2672 MOVL S^#SS$_NORMAL,R0 ; Supply a standard status for exit
04 147D 2673 RET ; Resume processing (or goto RMS_ERROR)
147E 2674 50$:
0769' CF 59 D0 147E 2675 MOVL R9,STATUS ; Save the status
58 D4 1483 2676 CLRL R8 ; Assume for now it's not SS failure
59 00000000' 8F D1 1485 2677 CMPL #SS$_SSFAIL,R9 ; But is it a System Service failure?
38 12 148C 2678 BNEQ 70$ ; BR if not - no special case message
148E 2679 $GETMSG_S MSGID = R10,- ; Get SS failure code associated text
148E 2680 MSGLEN = BUFFER_PTR,-
148E 2681 BUFADR = FAO_BUF,-
148E 2682 FLAGS = #14,-
148E 2683 OUTADR = MSG_BLOCK
079A' CF 95 14A5 2684 TSTB MSG_BLOCK+1 ; Get FAO arg count for SS failure code
16 13 14A9 2685 BEQL 60$ ; Don't use $GETMSG if no $FAO args...
0053' CF DF 14AB 2686 PUSHAL BUFFER_PTR ; ...else build up...

```


00741130	01	DD	14AF	2687	PUSHL	#1	; ...a message describing...
	8F	DD	14B1	2688	PUSHL	#UETPS TEXT	; ...why the System Service failed
00	5A	F0	14B7	2689	INSV	R10,#STSSV SEVERITY,-	; Give the message...
6E	03		14BA	2690		#STSS SEVERITY,(SP)	; ...the correct severity code
58	03	D0	14BC	2691	MOVL	#3,R8	; Count the number of args we pushed
	05	11	14BF	2692	BRB	70\$	
			14C1	2693			
	5A	DD	14C1	2694	PUSHL	R10	; Save SS failure code
58	01	D0	14C3	2695	MOVL	#1,R8	; Count the number of args we pushed
			14C6	2696			
57	66	04	C5	14C6	MULL3	#4,CHF\$SIG_ARGS(R6),R7	; Convert longwords to bytes
	5E	57	C2	14CA	SUBL2	R7,SP	; Save the current signal array...
6E	04	A6	57	28	MOVC3	R7,CHF\$SIG_NAME(R6),(SP)	; ...on the stack
7E	66	58	C1	14D2	ADDL3	R8,CHF\$SIG_ARGS(R6),-(SP)	; Push the current arg count
	00A6	31	14D6	2701	BRW	ERROR_EXIT	


```

14D9 2703 .SBTTL RMS Error Handler
14D9 2704 :++
14D9 2705 : FUNCTIONAL DESCRIPTION:
14D9 2706 : This routine handles error returns from RMS calls.
14D9 2707 :
14D9 2708 : CALLING SEQUENCE:
14D9 2709 : Called by RMS when a file processing error is found.
14D9 2710 :
14D9 2711 : INPUT PARAMETERS:
14D9 2712 : NONE
14D9 2713 :
14D9 2714 : IMPLICIT INPUTS:
14D9 2715 : The FAB or RAB associated with the RMS call.
14D9 2716 :
14D9 2717 : OUTPUT PARAMETERS:
14D9 2718 : NONE
14D9 2719 :
14D9 2720 : IMPLICIT OUTPUTS:
14D9 2721 : Error message
14D9 2722 :
14D9 2723 : COMPLETION CODES:
14D9 2724 : NONE
14D9 2725 :
14D9 2726 : SIDE EFFECTS:
14D9 2727 : Program may exit, depending on severity of the error.
14D9 2728 :
14D9 2729 :--
14D9 2730
14D9 2731 RMS_ERROR:
14D9 2732 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
14DB 2733
14DB 2734 MOVL 4(AP),R6 ; See whether we're dealing with...
14DF 2735 CMPB #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
14E2 2736 BNEQ 10$ ; BR if it's a RAB
14E4 2737 MOVAL FILE,R7 ; FAB-specific code: text string...
14E9 2738 MOVL R6,R8 ; ...address of FAB...
14EC 2739 PUSHL FAB$STV(R6) ; ...STV field for error...
14EF 2740 PUSHL FAB$STS(R6) ; ...STS field for error...
0769'CF 08 A6 DD 14F2 2741 MOVL FAB$STS(R6),STATUS ; ...and save the error code
15 11 14F8 2742 BRB COMMON ; FAB and RAB share other code
14FA 2743 10$:
14FA 2744 MOVAL RECORD,R7 ; RAB-specific code: text string...
14FF 2745 MOVL RAB$FAB(R6),R8 ; ...address of associated FAB...
1503 2746 PUSHL RAB$STV(R6) ; ...STV field for error...
1506 2747 PUSHL RAB$STS(R6) ; ...STS field for error...
0769'CF 08 A6 DD 1509 2748 MOVL RAB$STS(R6),STATUS ; ...and save the error code
150F 2749 COMMON:
150F 2750 MOVZBL FAB$B_FNS(R8),R10 ; Get the file name size
1513 2751 $FAO_S CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
1513 2752 OUTLEN = BUFFER_PTR,-
1513 2753 OUTBUF = FAO_BUF,-
1513 2754 P1 = R7,-
1513 2755 P2 = R10,-
1513 2756 P3 = FAB$FNA(R8)
0053'CF DF 152D 2757 PUSHAL BUFFER_PTR ; ...and arguments for ERROR_EXIT...
01 DD 1531 2758 PUSHL #1 ; ...
00741130 8F DD 1533 2759 PUSHL #UETP$TEXT ; ...

```


59	00	EF	1539	2760	EXTZV	#STS\$V-SEVERITY,-	
	03		153B	2761		#STS\$S-SEVERITY,-	
	05		153C	2762		STATUS,R9	
	6E	88	1540	2763	BISB2	R9,(SP)	; ...get the severity code...
	05	DD	1543	2764	PUSHL	#5	; ...and add it into the signal name
	0037	31	1545	2765	BRW	ERROR_EXIT	; Current arg count


```

1548 2767 .SBTTL CTRL/C Handler
1548 2768 :++
1548 2769 : FUNCTIONAL DESCRIPTION:
1548 2770 : This routine handles CTRL/C AST's
1548 2771 :
1548 2772 : CALLING SEQUENCE:
1548 2773 : Called via AST
1548 2774 :
1548 2775 : INPUT PARAMETERS:
1548 2776 : NONE
1548 2777 :
1548 2778 : IMPLICIT INPUTS:
1548 2779 : NONE
1548 2800 :
1548 2801 : OUTPUT PARAMETERS:
1548 2802 : NONE
1548 2803 :
1548 2804 : IMPLICIT OUTPUTS:
1548 2805 : NONE
1548 2806 :
1548 2807 : COMPLETION CODES:
1548 2808 : NONE
1548 2809 :
1548 2810 : SIDE EFFECTS:
1548 2811 : NONE
1548 2812 :
1548 2813 :--
1548 2814 :
1548 2815 CCASTHAND:
1548 2816 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
1548 2817
1548 2818 PUSHAL CNTRLCMSG ; Set message pointer
1548 2819 PUSHAL #1 ; Set arg count
1548 2820 PUSHAL #UETP$_TEXT!ST$K_WARNING ; Set signal name
1548 2821 PUSHAL #0 ; Indicate an abnormal termination
1548 2822 PUSHAL PROCESS_NAME ; ...
1548 2823 PUSHAL #2 ; ...
1548 2824 PUSHAL #UETP$_ABEND!ST$K_WARNING ; ...
1548 2825 CALLS #7,G^LIB$SIGNAL ; Output the message
1548 2826 MOVL #<ST$M_INHIB_MSG!- ; Set the exit status
1548 2827 SS$ CONTROLC=-
1548 2828 ST$K_SUCCESS+ST$K_WARNING>,-
1548 2829 STATUS
1548 2830 $EXIT_S STATUS ; Terminate program cleanly

```

```

02CF'CF DF
00741130 8F DD
00 00 DD
06C3'CF DF
02 02 DD
007410E0 8F DD
00000000'GF 07 FB
DO
0769'CF OFFFFFFFFF'8F

```



```

157F 2812 .SBTTL Error Exit
157F 2813 :++
157F 2814 : FUNCTIONAL DESCRIPTION:
157F 2815 : This routine prints an error message and exits.
157F 2816 :
157F 2817 : CALLING SEQUENCE:
157F 2818 : MOVx error status value,STATUS
157F 2819 : PUSHx error specific information on the stack
157F 2820 : PUSHL current argument count
157F 2821 : BRW ERROR_EXIT
157F 2822 :
157F 2823 : INPUT PARAMETERS:
157F 2824 : Arguments to LIB$SIGNAL, as above
157F 2825 :
157F 2826 : IMPLICIT INPUTS:
157F 2827 : NONE
157F 2828 :
157F 2829 : OUTPUT PARAMETERS:
157F 2830 : Message to SYS$OUTPUT and SYS$ERROR
157F 2831 :
157F 2832 : IMPLICIT OUTPUTS:
157F 2833 : Program exit
157F 2834 :
157F 2835 : COMPLETION CODES:
157F 2836 : NONE
157F 2837 :
157F 2838 : SIDE EFFECTS:
157F 2839 : NONE
157F 2840 :
157F 2841 :--
157F 2842 :
157F 2843 ERROR_EXIT:
157F 2844
157F 2845 $SETAST_S ENBFLG = #0 ; ASTs can play havoc with messages
157F 2846 BBS #BEGIN_MSGV,FLAG,10$ ; BR if "begin" msg already printed
157F 2847 CLRL -(SP) ; Set the time stamp flag
157F 2848 PUSHAL TEST_NAME ; Set the test name
157F 2849 PUSHL #2 ; Push the argument count
157F 2850 PUSHL #UETP$_BEGIN!ST$K_SUCCESS ; Set the message code
157F 2851 CALLS #4,G^LIB$SIGNAL ; Print the startup message
157F 2852 10$:
157F 2853 ADDL3 (SP)+,#8,ARG_COUNT ; Get total # args, pop partial count
157F 2854 INCL ERROR_COUNT ; Keep running error count
157F 2855 PUSHL #0 ; Push the time parameter
157F 2856 PUSHAL PROCESS_NAME ; Push test name...
157F 2857 PUSHL #^XF0002 ; ...arg count...
157F 2858 PUSHL #UETP$_ABEND!ST$K_ERROR ; ...and signal name
157F 2859 PUSHL ERROR_COUNT ; Finish off arg list...
157F 2860 PUSHAL PROCESS_NAME ; ...
157F 2861 PUSHL #^X10002 ; ...
157F 2862 PUSHL #UETP$_ERBOXPROC!ST$K_ERROR ; ...for error box message
157F 2863 CALLS ARG_COUNT,G^LIB$SIGNAL ; Truly bitch
157F 2864 15DC 2864
157F 2865 TSTL STATUS ; Did we exit with an error code?
157F 2866 BNEQ 20$ ; BR if we did
157F 2867 MOVL #UETP$_ABEND!ST$K_ERROR,- ; Supply a generic one otherwise
157F 2868 STATUS

```


UETUNAS00
V04-000

VAX/VMS UETP DEVICE TEST FOR THE UNA J 12
Error Exit

16-SEP-1984 01:37:49 VAX/VMS Macro V04-00
5-SEP-1984 04:26:50 [UETP.SRC]UETUNAS00.MAR;1

Page 69
(30)

0769'CF 10000000 8F C8 15EB 2869 20\$:
15EB 2870
15F4 2871

BISL #STSSM-INHIB_MSG,STATUS ; Don't print messages twice!
\$EXIT_S STATUS ; Exit in error


```

15FF 2873 .SBTTL Exit Handler
15FF 2874 :++
15FF 2875 : FUNCTIONAL DESCRIPTION:
15FF 2876 : This routine handles cleanup at exit. If the MODE logical name is
15FF 2877 : equated to "ONE", the routine will update the test flag in the
15FF 2878 : UETINIDEV.DAT file depending on the UETUNTSM_TESTABLE flag state in the
15FF 2879 : UETUNT$B_FLAGS field of the unit block for each unit for the device
15FF 2880 : under test.
15FF 2881 :
15FF 2882 : CALLING SEQUENCE:
15FF 2883 : Invoked automatically by $EXIT System Service.
15FF 2884 :
15FF 2885 : INPUT PARAMETERS:
15FF 2886 : STATUS contains the exit status.
15FF 2887 : FLAG has synchronizing bits.
15FF 2888 : DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
15FF 2889 :
15FF 2890 : IMPLICIT INPUTS:
15FF 2891 : UNIT_LIST points to the head of a doubly linked circular list of unit
15FF 2892 : blocks for the device under test.
15FF 2893 :
15FF 2894 : OUTPUT PARAMETERS:
15FF 2895 : NONE
15FF 2896 :
15FF 2897 : IMPLICIT OUTPUTS:
15FF 2898 : Various files are de-accessed, the process name is reset, and any
15FF 2899 : necessary synchronization with UETPDEV01 is carried out.
15FF 2900 : If the MODE logical name is equated to "ONE", the routine will update
15FF 2901 : the test flag in the UETINIDEV.DAT file depending on the
15FF 2902 : UETUNTSM_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
15FF 2903 : block for each unit for the device under test.
15FF 2904 :
15FF 2905 : COMPLETION CODES:
15FF 2906 : NONE
15FF 2907 :
15FF 2908 : SIDE EFFECTS:
15FF 2909 : NONE
15FF 2910 :
15FF 2911 :--
15FF 2912 :
15FF 2913 EXIT_HANDLER:
OFFC 15FF 2914 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
1601 2915
1601 2916 $SETSFM_S ENBFLG = #0 ; Turn off System Service failure mode
160A 2917 $SETAST_S ENBFLG = #0 ; We're finished - no more ASTs
1613 2918 $TRNLOG_S LOGNAM = MODE, - ; Get the run mode
1613 2919 RSLLEN = BUFFER_PTR, -
1613 2920 RSLBUF = FAO BUF
162C 2921 BICB2 #LC BITM,BUFFER ; Convert to upper case
005B'CF 20 8A 1631 2922 CMPB #^A70/,BUFFER ; Is this a one shot?
005B'CF 4F 8F 91 1637 2923 BEQL 10$ ; BR if yes...
00B8 31 1639 2924 BRW END_UPDATE ; ...else don't update UETINIDEV.DAT
03 0049'CF 02 E0 163C 2925 10$: BBS #SAFE TO UPDV,FLAG,20$ ; Only update if it's safe
00AF 31 1642 2927 BRW END_UPDATE ; Else forget it
5A 0E40'CF DE 1645 2928 20$: MOVAL INI_RAB,R10 ; Set the RAB address
1645 2929

```



```

10 AA 0E84'CF 02 90 164A 2930      MOVB  #RAB$C_RFA,RAB$B_RAC(R10) ; Set RFA mode
                                MOVBC3 #6,DDB_RFA,RAB$W_RFA(R10) ; Set RFA to DDB line
                                $GET  RAB = (R10) ; Go back to the DDB record
                                BLBC  R0,UPDATE_FAILED ; If failure then forget it
5B 07B8'CF 000007B8'8F 00 90 1661 2934      MOVB  #RAB$C_SEQ,RAB$B_RAC(R10) ; Set back to sequential mode
                                ADDL3 #UNIT_LIST,UNIT_LIST,R11 ; Set the unit block list header
                                CLRL  R9 ; Init a counter
                                UNIT_LOOP:
                                BBC  #UETUNT$V_TESTABLE,- ; BR if this unit is not testable
                                UETUNT$B_FLAGS(R11),10$ ;
                                INCL  R9 ; Count testable units
                                10$:
                                ADDL2 (R11),R11 ; Next unit block
                                CMPL  R11,#UNIT_LIST ; Are we full circle in the list?
                                BNEQ  UNIT_LOOP- ; BR if not
                                TSTL  R9 ; Any testable units?
                                BNEQ  20$ ; BR if yes...
                                MOVB  #^A/N/,BUFFER+4 ; ...else disable the DDB record...
                                $UPDATE RAB = (R10) ;
                                BLBC  R0,UPDATE_FAILED ; If error then forget it
                                20$:
                                ADDL2 (R11),R11 ; Next unit block
                                CMPL  R11,#UNIT_LIST ; Are we full circle in the list?
                                BEQL  END_UPDATE ; BR if yes
                                $GET  RAB = (R10) ; Get a record
                                BLBC  R0,UPDATE_FAILED ; If error then forget it
                                BICB2 #LC_BITM,BUFFER ; Convert to uppercase
                                CMPB  #^A7U/,BUFFER ; Is it a UCB record?
                                BNEQ  END_UPDATE ; BR if not
                                BBS  #UETUNT$V_TESTABLE,- ; BR if this unit is testable...
                                UETUNT$B_FLAGS(R11),20$ ;
                                MOVB  #^A/N/,BUFFER+4 ; ...else disable the UCB record...
                                $UPDATE RAB = (R10) ;
                                BLBS  R0,20$ ; ...here
                                UPDATE_FAILED: ; Look at the next record if no error
                                PUSHL  RAB$L_STV(R10) ; Do a simple message...
                                PUSHL  R0 ; ...to tell of the failure
                                PUSHAL INIDEV_UPDERR
                                PUSHL  #1
                                EXTZV #STSS$V_SEVERITY,- ; Copy the severity from RMS status...
                                #STSS$SEVERITY,R0,-(SP) ;
                                BISL2 #UETP$TEXT,(SP) ; ...to our message
                                CALLS #5,G^LIB$SIGNAL
                                END_UPDATE:
                                PUSHL  #0 ; Set the time flag
                                PUSHAL TEST_NAME ; Push the test name
                                PUSHL  #2 ; Push arg count
                                EXTZV #STSS$V_SEVERITY,- ; Push the proper exit severity...
                                #STSS$SEVERITY,-
                                STATUS,-(SP) ;
                                BISL2 #UETP$_ENDEDD,(SP) ; ...and use it in our message code
                                PUSHL  #4 ;
                                MOVL  SP,R1 ;
                                $PUTMSG_S MSGVEC = (R1) ; Output the message
                                $SETPRN_S PRCNAM = ACNT_NAME ; Reset the process name
                                RET ; That's all folks!
                                04 1729 2985
                                172A 2986

```


UETUNAS00
V04-000

VAX/VMS UETP DEVICE TEST FOR THE UNA^{M 12}
Exit Handler

16-SEP-1984 01:37:49
5-SEP-1984 04:26:50

VAX/VMS Macro V04-00
[UETP.SRC]UETUNAS00.MAR;1

Page 72
(31)

172A 2987

.END

UETUNAS00

-S
Vi
St
Im
Im
Im
Nu
Nu
Nu
Nu
Nu
Nu
Us
Nu
Im
Ma
Es
Pe
--
To
Us
To
Nu
6
A
LI

UETUNAS00
Symbol table

VAX/VMS UETP DEVICE TEST FOR THE UNA

N 12

16-SEP-1984 01:37:49 VAX/VMS Macro V04-00
5-SEP-1984 04:26:50 [UETP.SRC]UETUNAS00.MAR;1

Page 73
(31)

```

$$TAB          = 00000F2C R    03
$$TABEND       = 00000F70 R    03
$$TMP          = 00000000
$$TMP1         = 00000001
$$TMP2         = 0000006A
$$TMPX         = 00000016 R    04
$$TMPX1        = 0000000D
$$T1           = 00000000
$$T2           = 00000006
ABORT_MSG      = 000007A2 R    02
ABRT_BRDCST    = 00000D26 R    05
ACNT_NAME      = 00000000 R    02
ADR            = 0000087E R    03
ADR1           = 000008DC R    03
ADR2           = 000008EA R    03
ALL_SET        = 000003F7 R    05
ARG_COUNT      = 000007AD R    03
ARG_LIST       = 0000081C R    03
ASTPAR_ERRMSG  = 00000A77 R    02
BADLEN         = 0000022A R    02
BEGIN_MSGM     = 00000008
BEGIN_MSGV     = 00000003
BRDCST_BUF     = 000008F8 R    03
BRDCST_BUF_LEN = 0000002E
BRDCST_NEEDED  = 00000040
BRDCST_NEDVD  = 00000006
BUFFER         = 0000005B R    03
BUFFER_PTR     = 00000053 R    03
CAB1_UNPLUGED  = 0000037D R    02
CAB2_UNPLUGED  = 000003AA R    02
CASE_FAILED    = 0000029A R    02
CCASTHAND      = 00001548 R    05
CHECK_IOSB     = 00001057 R    05
CHFSL_SIGARGLST = 00000004
CHFSL_SIG_ARG1 = 00000008
CHFSL_SIG_ARGS = 00000000
CHFSL_SIG_NAME = 00000004
CHK_LOOP       = 000012EA R    05
CHK_QIO_AST    = 00001070 R    05
CLEAN_EXIT     = 00000F37 R    05
CNTRLMSG       = 000002CF R    02
CNTR_TBL       = 0000088D R    02
COMMON         = 0000150F R    05
CONTROLLER     = 00000031 R    02
CONT_DESC      = 000009DE R    02
COUNTER        = 00000CBDE R    03
COUNTER_CHECK  = 000012D1 R    05
COUNTER_EXIT   = 000013D0 R    05
COUNTER_MSG    = 00000271 R    02
CS1            = 000000A1 R    02
CS3            = 000000B3 R    02
CVTRSL        = 00000814 R    03
DDB_RFA        = 00000E84 R    03
DEAD_CTRLNAME  = 00000310 R    02
DEV$V_TRM      = ***** X 05
DEVDEP_SIZE    = 00000000
DEVVSC         = 000006BB R    03

```

```

DEVNAM_LEN     = 00000787 R    03
DEV_NAME       = 000006DA R    03
DIAG_BUF       = 00000C00 R    03
DIAG_BUF_SIZE  = 00000050
DIB            = 000006E9 R    03
DIB$B_DEVCLASS = 00000004
DIB$B_DEVTYPE  = 00000005
DIB$K_LENGTH   = 00000074
DIBBUF         = 000006F1 R    03
DO_BRDCST      = 000009A8 R    05
DUMMY_FAB      = 00000EDC R    03
DUMMY_RAB      = 00000F2C R    03
DUMP           = 000000E7 R    02
DUMP_MODEM     = 00000001
DUMP_MODEV     = 00000000
DUMP_SIGNAL    = 000012BD R    05
DUMP_STAMPS    = 00001256 R    05
DVIS_DEVNAM    = 00000020
ECHO_SWITCH    = 0000086E R    03
EFN2           = 00000004
END_ADR        = 00000BE2 R    03
END_UPDATE     = 000016F4 R    05
ERRCNT_BUF     = 000009DE R    03
ERRCNT_LEN     = 00000080
ERRCOUNT_DESC = 000009D6 R    03
ERROR_COUNT    = 00000765 R    03
ERROR_EXIT     = 0000157F R    05
ERROR_TEST     = 00000DAF R    05
ERRTEST_MSG    = 00000A5B R    02
ERRTST_ERR     = 00000E51 R    05
ERRTST_P2BUF   = 000009CE R    03
ERRTST_P2DESC  = 000009C6 R    03
ERRTST_P2LEN   = 00000008
EXIT_DESC      = 0000079D R    03
EXIT_HANDLER   = 000015FF R    05
FAB$B_BID      = 00000000
FAB$B_FNS      = 00000034
FAB$C_BID      = 00000003
FAB$C_BLN      = 00000050
FAB$C_SEQ      = 00000000
FAB$C_VAR      = 00000002
FAB$L_ALQ      = 00000010
FAB$L_DEV      = 00000040
FAB$L_FNA      = 0000002C
FAB$L_FOP      = 00000004
FAB$L_STS      = 00000008
FAB$L_STV      = 0000000C
FAB$V_CHAN_MODE = 00000002
FAB$V_CR       = 00000001
FAB$V_FILE_MODE = 00000004
FAB$V_GET      = 00000001
FAB$V_LNM_MODE = 00000000
FAB$V_PUT      = 00000000
FAB$V_UFO      = 00000011
FAB$V_UPD      = 00000003
FAB$V_UPI      = 00000006
FAB$W_GBC      = 00000048

```


UETUNAS00
Symbol table

VAX/VMS UETP DEVICE TEST FOR THE UNA B 13

16-SEP-1984 01:37:49 VAX/VMS Macro V04-00
5-SEP-1984 04:26:50 [UETP.SRC]UETUNAS00.MAR;1

Page 74
(31)

FAIL_OUT	00000FE6	R	05
FAO_BUF	0000004B	R	03
FILE	000009E6	R	02
FIND_IT	000001E9	R	05
FLAG	00000049	R	03
FLAG_SHUTDNM	= 00000010		
FLAG_SHUTDNV	= 00000004		
FORWARD_MSG_SIZ	= 0000000E		
FOUND_IT	00000285	R	05
GETSYT ITMLST	00000AD1	R	02
HALFMIN	000009BE	R	02
HARD_ADR	= 0000002A		
ID_CTRSTR	00000180	R	02
ID_FNDM	= 00000020		
ID_FNDV	= 00000005		
ILLEGAL_REC	000003DA	R	02
INADDRESS	00000775	R	03
INIDEV_UPDERR	0000C0C2	R	02
INI_FAB	00000DF0	R	03
INI_RAB	00000E40	R	03
INPUT ITMLST	00000083	R	02
IOSM_CLR_COUNT	*****	X	05
IOSM_CTRL	*****	X	05
IOSM_CTRLCAST	*****	X	05
IOSM_NOW	*****	X	05
IOSM_RD_COUNT	*****	X	05
IOSM_SHUTDOWN	*****	X	05
IOSM_STARTUP	*****	X	05
IOS_READLBLK	*****	X	05
IOS_READVBLK	*****	X	05
IOS_SENSEMODE	*****	X	05
IOS_SETCHAR	*****	X	05
IOS_SETMODE	*****	X	05
IOS_WRITEBLK	*****	X	05
IOS_WRITEVBLK	*****	X	05
ITERATION	00000791	R	03
LC_BITM	= 00000020		
LIB\$CVT HTB	*****	X	05
LIB\$SIGNAL	*****	X	05
LIN	= 00000000		
LOGNAM	0000004D	R	02
LONGORSHRTM	= 00000080		
LONGORSHRTV	= 00000007		
LOOPBACK_ACK	00C00C54	R	03
LOOPBACK_AST	000010F1	R	05
LOOPBACK_FAIL_MSG	00000667	R	02
LOOPBACK_TMO	00001087	R	05
LOOPBACK_TMO_MSG	00000723	R	02
LOOP_PROTOCOL	= 00000090		
MAXBUF	00000C50	R	03
MAXBUF MSG	00000AE1	R	02
MAX_DEV_DESIG	= 0000000A		
MAX_PROG_NAME	= 0000000F		
MAX_UNABOF SIZE	= 000005DC		
MAX_UNIT_DESIG	= 00000005		
MIN_MAXBUF	= 00000630		
MODE	00000041	R	02

MSG_BLOCK	00000799	R	03
NAME_LEN	= 0000000F		
NEW_NODE	000007C0	R	03
NIADRTESTING	0000013C	R	02
NIADRWRONG	000000F3	R	02
NMASC_CTLIN_BSM	= 000003F7		
NMASC_CTLIN_CDC	= 00000425		
NMASC_CTLIN_LBE	= 00000411		
NMASC_CTLIN_OVR	= 00000428		
NMASC_CTLIN_RFL	= 00000426		
NMASC_CTLIN_SBU	= 00000429		
NMASC_CTLIN_UBU	= 0000042A		
NMASC_LINCN_NOR	= 00000000		
NMASC_LINMC_SET	= 00000001		
NMASC_PCLI_BFN	= 00000451		
NMASC_PCLI_BUS	= 00000AF1		
NMASC_PCLI_CON	= 00000456		
NMASC_PCLI_CRC	= 00000B1C		
NMASC_PCLI_DCH	= 00000B1B		
NMASC_PCLI_EKO	= 00000B1F		
NMASC_PCLI_MCA	= 00000B0F		
NMASC_PCLI_MLT	= 00000B19		
NMASC_PCLI_PAD	= 00000B1A		
NMASC_PCLI_PRM	= 00000B18		
NMASC_PCLI_PTY	= 00000B0E		
NMASC_STATE_OFF	= 00000001		
NMASC_STATE_ON	= 00000000		
NMASS_CNT_TYP	= 0000000C		
NMASS_CNT_WID	= 00000002		
NMASV_CNT_MAP	= 0000000C		
NMASV_CNT_TYP	= 00000000		
NMASV_CNT_WID	= 0000000D		
NOUNIT_SELECTED	00000357	R	02
NO_CTRNAME	000002F0	R	02
NO_RMS_AST_TABLE	0000005E	R	02
NRAT_LENGTH	= 00000014		
OT\$SCVT TI_L	*****	X	05
OUTADDRESS	0000077D	R	03
OUT_OF_SEQ_MSG	0000075B	R	02
P1BUF	00000834	R	03
P2_DESC	00000884	R	03
P2_DESC1	000008D4	R	03
P2_PARAM	0000083C	R	03
P2_PARAM1	0000088C	R	03
PAD_SWITCH	00000862	R	03
PAGES	= 00000007		
PASS	00000795	R	03
PASS_MSG	0000085A	R	02
PC1...	= 000008B7	R	02
PC2...	= 0000098E	R	02
PCSR1	= 0000001A		
PCSR1\$S_SLFTST	= 00000006		
PCSR1\$V_ICAB	= 0000000E		
PCSR1\$V_SLFTST	= 00000008		
PCSR1\$V_XPWR	= 0000000F		
PMTSIZ	= 00000019		
PRM	= 00000064		

UETUNAS00
Symbol table

VAX/VMS UETP DEVICE TEST FOR THE UNA C 13

16-SEP-1984 01:37:49 VAX/VMS Macro V04-00
5-SEP-1984 04:26:50 [UETP.SRC]UETUNAS00.MAR;1

Page 75
(31)

PROCESS_NAME	= 000006C3	R	03	SAFE_TO_UPDM	= 00000004		
PROCESS_NAME FREE	= 0000000B			SAFE-TO-UPDV	= 00000002		
PROC_CONT_NAME	00000093	R	05	SAVE_STAT	00000BF6	R	03
PROMPT	00000A21	R	02	SBP	00000C58	R	03
PROTOCOL	00000868	R	03	SEC\$M_EXPREG	*****	X	05
QUAD STATUS	0000076D	R	03	SEC\$M_GBL	*****	X	05
RAB\$B_PSZ	= 00000034			SELF_ADR	000008FC	R	03
RAB\$B_RAC	= 0000001E			SENSE_ERRMSG	00000A3A	R	02
RAB\$C_BID	= 00000001			SENSE_P1BUF	00000926	R	03
RAB\$C_BLN	= 00000044			SENSE_P2BUF	00000936	R	03
RAB\$C_RFA	= 00000002			SENSE_P2DESC	0000092E	R	03
RAB\$C_SEQ	= 00000000			SENSE_P2LEN	= 00000090		
RAB\$C_CTX	= 00000018			SENSE_TEST	00000D3C	R	05
RAB\$C_FAB	= 0000003C			SHR\$_ABENDD	= 000010E0		
RAB\$C_PBF	= 00000030			SHR\$_BEGINDD	= 00001038		
RAB\$C_ROP	= 00000004			SHR\$_ENDEDD	= 00001080		
RAB\$C_STS	= 00000008			SHR\$_TEXT	= 00001130		
RAB\$C_STV	= 0000000C			SIZE-TBL	0000098E	R	02
RAB\$V_PMT	= 0000001E			SIZE-TBL1	0000099E	R	02
RAB\$W_RFA	= 00000010			SIZE-TBL SIZE	= 00000004		
RAB\$W_RSZ	= 00000022			SLFTST_FAILED	00000A9A	R	02
RANDOM1	00000789	R	03	SS\$_ABORT	*****	X	05
RANDOM2	0000078D	R	03	SS\$_BADPARAM	*****	X	05
RCV_IOSB	00000BEE	R	03	SS\$_BUFFEROVF	*****	X	05
RDID_EFN	= 00000007			SS\$_CONTROL	*****	X	05
READ_ERRCOUNT	00000EC7	R	05	SS\$_DATAOVERUN	*****	X	05
READ_SIZE	= 000005DC			SS\$_ENDOFFILE	*****	X	05
RECORD	000009F2	R	02	SS\$_IVBUFLN	*****	X	05
RECVPOOL_SIZ	= 00000004			SS\$_NORMAL	*****	X	05
RECV_AST	00001182	R	05	SS\$_NOSUCHSEC	*****	X	05
RECV_BUF_ADR	00000BFC	R	03	SS\$_NOTRAN	*****	X	05
RECV_EFN	= 00000006			SS\$_SSFAL	*****	X	05
RECV_STAMP	00001221	R	05	SS\$_TIMEOUT	*****	X	05
REC_SIZE	= 00000028			SS\$_WASSET	*****	X	05
REMOTE_ID_CTR	000001BE	R	02	SSERROR	000013F6	R	05
REMOTE_LOOPBACK_TEST	00000BC2	R	05	SS SYNCH EFN	= 00000003		
REMOTE_NOTFND	00000200	R	02	STAMP_BUFFER	00000C5C	R	03
REPORT	00000093	R	02	START_CONT	000007DA	R	05
REQUEST_BUF	000000DF	R	03	STATUS	00000769	R	03
REQUEST_BUFSIZ	= 000005DC			STR\$UPCASE	*****	X	05
REQUEST_FND	00000E58	R	05	ST\$K_ERROR	= 00000002		
REQUEST_NOTFND	00000EBD	R	05	ST\$K_INFO	= 00000003		
RESTART	0000085A	R	05	ST\$K_SEVERE	= 00000004		
RETRANS_MSG	000006E1	R	02	ST\$K_SUCCESS	= 00000001		
RETRY_COUNT	= 00000004			ST\$K_WARNING	= 00000000		
RETRY_MSG	000005E9	R	02	ST\$M_INHIB MSG	= 10000000		
RMSS_BLN	*****	X	02	ST\$S_FAC NO	= 0000000C		
RMSS_BUSY	*****	X	02	ST\$S_SEVERITY	= 00000003		
RMSS_CDA	*****	X	02	ST\$V_FAC NO	= 00000010		
RMSS_FAB	*****	X	02	ST\$V_SEVERITY	= 00000000		
RMSS_FACILITY	= 00000001			SUC_EXIT	00000F60	R	05
RMSS_RAB	*****	X	02	SUPDEV_GBLSEC	00000020	R	02
RMS_ERROR	000014D9	R	05	SUP_FAB	00000E8C	R	03
RMS_ERR_STRING	00000A00	R	02	SYIS_MAXBUF	= 0000104F		
RSLBUF	000007CC	R	03	SY\$S_ASSIGN	*****	GX	05
RSLEN	000007C8	R	03	SY\$S_CANCEL	*****	GX	05
RW_TIME_ID	= 00000003			SY\$S_CANTIM	*****	GX	05

UETUNAS00
Symbol table

VAX/VMS UETP DEVICE TEST FOR THE UNA

D 13

16-SEP-1984 01:37:49 VAX/VMS Macro V04-00
5-SEP-1984 04:26:50 [UETP.SRC]UETUNAS00.MAR;1

Page 76
(31)

SYSS\$CLREF	*****	GX	05	TIME_SUC_OUT	000013EE	R	05
SYSS\$CONNECT	*****	GX	05	TSCLE	= 00000100		
SYSS\$CRMPSC	*****	GX	05	TTCHAN	00000000	R	03
SYSS\$DASSGN	*****	GX	05	TTNAME	= 0000000A	R	03
SYSS\$DCLEXH	*****	GX	05	TTNAME_LEN	= 00000009		
SYSS\$EXIT	*****	GX	05	TTNAME-ROPTR	000002C7	R	02
SYSS\$EXPREG	*****	GX	05	TTNAME-RWPTR	00000002	R	03
SYSS\$FAO	*****	X	05	TWENSEC	000009B6	R	02
SYSS\$FAOL	*****	GX	05	TWOMIN	000009C6	R	02
SYSS\$GET	*****	GX	05	UETP	= 00740000		
SYSS\$GETDEV	*****	GX	05	UETPS_ABEND	= 007410E0		
SYSS\$GETDVI	*****	GX	05	UETPS_ABORTC	= 0074832B		
SYSS\$GETMSG	*****	GX	05	UETPS-BEGIN	= 00741038		
SYSS\$GETSYI	*****	GX	05	UETPS-DATAEVERR	= 00748018		
SYSS\$GETTIM	*****	GX	05	UETPS-DENOSU	= 00748333		
SYSS\$INPUT	00000072	R	02	UETPS-DEUNUS	= 0074819A		
SYSS\$MGBLSC	*****	GX	05	UETPS-ENDED	= 00741080		
SYSS\$OPEN	*****	GX	05	UETPS-ERBOXPROC	= 00748020		
SYSS\$PUTMSG	*****	GX	05	UETPS-FACILITY	= 00000074		
SYSS\$QIO	*****	GX	05	UETPS-TEXT	= 00741130		
SYSS\$QIOW	*****	GX	05	UETUNAS00	00000000	RG	05
SYSS\$READER	*****	GX	05	UETUNT\$B_FLAGS	= 0000000B		
SYSS\$SETAST	*****	GX	05	UETUNT\$B_TYPE	= 00000008		
SYSS\$SETEF	*****	GX	05	UETUNT\$C_FAB	= 00000110		
SYSS\$SETIMR	*****	GX	05	UETUNT\$C_INDSIZ	= 000001A4		
SYSS\$SETPRN	*****	GX	05	UETUNT\$K_DEVDEP	= 000001A4		
SYSS\$SETSF	*****	GX	05	UETUNT\$K_FAB	= 00000110		
SYSS\$TRNLOG	*****	GX	05	UETUNT\$K_RAB	= 00000160		
SYSS\$UPDATE	*****	GX	05	UETUNT\$M_TESTABLE	= 00000002		
SYSS\$WAITFR	*****	GX	05	UETUNT\$T_FILSPC	= 00000014		
SYSIN_FAB	00000D5C	R	03	UETUNT\$V_TESTABLE	= 00000001		
SYSIN_RAB	00000DAC	R	03	UETUNT\$W_SIZE	= 00000009		
TBL_END	000008B7	R	02	UNA_SHUTDOWN	0000102C	R	05
TBL_SIZE	= 00000007			UNA_STARTUP	00000FF8	R	05
TENSEC	000009AE	R	02	UNIT_DESC	000009D6	R	02
TEST_NAME	0000000F	R	02	UNIT_LIST	000007B8	R	03
TEST_OVERM	= 00000002			UNIT_LOOP	00001671	R	05
TEST_OVERV	= 00000001			UNIT_NUMBER	00000785	R	03
TEXT_BUFFER	= 00000084			UPDATE_FAILED	000016D6	R	05
THREEMIN	000009CE	R	02	WRITE_SIZE	= 000005DC		
TIMEOUT1	0000040E	R	02	XE_CHAN	00000818	R	03
TIMEOUT10	000005B1	R	02	XE_CHAN1	0000081A	R	03
TIMEOUT11	000005CB	R	02	XE_IOSB	00000BE6	R	03
TIMEOUT2	0000042F	R	02	XMSM_CHR_LOOPB	= 00000002		
TIMEOUT3	00000453	R	02	XMIT	00000878	R	05
TIMEOUT4	00000483	R	02	XMIT_BUF_ADR	00000BF8	R	03
TIMEOUT5	000004A6	R	02	XMIT_EFN	= 00000005		
TIMEOUT6	000004DC	R	02	XMIT_LOOP	0000085A	R	05
TIMEOUT7	0000050A	R	02	XMIT_SETUP	00000BAF	R	05
TIMEOUT8	00000541	R	02	XMIT_STAMP	0000121B	R	05
TIMEOUT9	0000056B	R	02	XMIT_START	00000951	R	05
TIME_ERR_OUT	000013D1	R	05				
TIME_ID_1	= 00000001						
TIME_ID_2	= 00000002						
TIME_IT	00000689	R	05				
TIME_STAMP_INTRO_MSG	0000081D	R	02				
TIME_STAMP_MSG	000007DB	R	02				

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	00000B47 (2887.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RWDATA	00000F70 (3952.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
\$RMSNAM	00000023 (35.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
UNAS	0000172A (5930.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.07	00:00:00.34
Command processing	145	00:00:00.68	00:00:03.89
Pass 1	1212	00:00:37.89	00:01:19.61
Symbol table sort	0	00:00:03.65	00:00:07.76
Pass 2	822	00:00:10.43	00:00:22.13
Symbol table output	0	00:00:00.36	00:00:00.66
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	2221	00:00:53.12	00:01:54.44

The working set limit was 2000 pages.
215843 bytes (422 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2485 non-local and 94 local symbols.
2987 source lines were read in Pass 1, producing 59 object records in Pass 2.
69 pages of virtual memory were used to define 62 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[UETP.OBJ]UETP.MLB;1	2
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	1
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	55
TOTALS (all libraries)	58

2722 GETS were required to define 58 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:UETUNAS00/OBJ=OBJ\$:UETUNAS00 MSRC\$:UETUNAS00/UPDATE=(ENH\$:UETUNAS00)+EXECML\$/LIB+LIB\$:UETP/LIB

0413 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY